МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

В.Ю. Щербина

КОНСТРУКТОРСЬКЕ ПРОЕКТУВАННЯ ОБЛАДНАННЯ

КОНСПЕКТ ЛЕКЦІЙ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів,
які навчаються за спеціальністю 133 «Галузеве машинобудування»,
спеціалізацією «Інжинірінг, комп'ютерне моделювання та
проектування обладнання виробництв полімерних і будівельних
матеріалів та виробів»

Київ
КПІ ім. Ігоря Сікорського
2018

Конструкторське проектування обладнання. Конспект лекцій [Електронний ресурс]: навчальний посібник для студентів які навчаються за освітньо-науковою програмою магістерської підготовки, спеціальності 133 «Галузеве машинобудування», спеціалізації «Інжинирінг, комп'ютерне моделювання та проектування обладнання виробництв полімерних і будівельних матеріалів та виробів» / В. Ю. Щербина КПІ ім. Ігоря Сікорського. – Київ: КПІ ім. Ігоря Сікорського, 2018. – 83 с.
Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 4 від 20.12.2018 р.) за поданням Вченої ради інженерно-хімічного факультету (протокол № 10 від 26.11.2018 р.)

Електронне мережне навчальне видання

КОНСТРУКТОРСЬКЕ ПРОЕКТУВАННЯ ОБЛАДНАННЯ

КОНСПЕКТ ЛЕКЦІЙ

Укладачі: *Щербина Валерій Юрійович*, доктор техн. наук, доцент

Відповідальний редактор *Гондлях О.В.,* доктор техн. наук, професор
Рецензенти: *Марчевський В.М.*, канд. техн. наук, професор

Призначення посібника – закріпити та поглибити теоретичний програмний матеріал курсу лекцій навчальної дисципліни «Конструкторське проектування обладнання» з метою вивчення спеціальних новітніх методів проектування для вдосконалення та підвищення ефективності обладнання, що проектується, забезпечення високої якості виконання проектно-конструкторських розробок, реалізації найбільш ефективних проектних рішень. Дисципліна орієнтує студентів на світовий сучасний рівень науково-технічного прогресу в галузі розробки обладнання хімічного машинобудування.

В посібнику розглянуті питання використання та розробки систем і підсистем автоматизованого конструкторського проектування, основи програмування на функціональній мові AutoLISP та інструментальні засоби створення графічного інтерфейсу в системі AutoCAD.

ЗМІСТ

ВСТУП

Основне завдання конструкторського проектування - це реалізація принципових схем, отриманих на етапі функціонального проектування. При цьому виконується конструювання окремих деталей, компоновка вузлів з деталей і конструктивних елементів, агрегатів з вузлів, після чого оформляється технічна документація на об'єкт проектування.

До графічних документів відносяться складальні креслення і деталювання, графіки структурних сіток кінематичних ланцюгів, циклограми і залежності для вибору параметрів режимів роботи машин, агрегатів і пристроїв, схеми структурні, функціональні та принципові (електричні, електронні, гідравлічні і т.д.). До завдань оформлення конструкторської документації входить виготовлення текстових і графічних документів. Текстові документи крім описової частини містять характеристики та паспортні дані вузлів і агрегатів, технічні умови на виготовлення, складання, налагодження та експлуатацію, специфікації і т.д.

Однією з найактуальніших проблем сучасного машинобудування є скорочення терміну проектування технологічного устаткування та поліпшення техніко-економічних характеристик. Це можливе при системному застосуванні засобів обчислювальної техніки, яке дозволяє не тільки покращити але й істотно збільшити продуктивність праці конструктора-проектувальника. Тому одним з основних завдань вищої школи є підвищення комп'ютерної грамотності і широке впровадження електронно-обчислювальної техніки з використанням числових систем у навчальному процесі, особливо на етапах проектно-конструкторської підготовки.

Завдяки досконалому вивченню графічних комп’ютерних систем конструктор може автоматизувати виконання проектно-конструкторських робіт, а також доопрацьовувати й адаптувати типові графічні системи під конкретні прикладні задачі конструювання з метою ефективного використання систем проектування обладнання.

ТЕМА 1 МІСЦЕ КОНСТРУКТОРСЬКОГО ПРОЕКТУВАННЯ В ЖИТТЄВОМУ ЦИКЛУ ВИРОБІВ

Будь-які вироби, у тому числі механічне обладнання, за час свого існування проходять ряд етапів, від ідеї створення до впровадження у виробництво й утилізації, що називається життєвим циклом виробу.

Проектування як етап цього циклу, передує виробництву й робить можливим виготовлення необхідної кількості виробів із заданими характеристиками, забезпечує отримання технічної документації, що повністю й однозначно описує всі відомості, необхідні й достатні для виготовлення виробів.

Проектування являє собою складний і творчий процес діяльності фахівця (проектувальника), інваріантний до різних типів виробів і складності. Проектування вимагає від проектувальника крім спеціальних, предметних знань, також знань методології, засобів і правил виконання проектних процедур. Сучасне проектування здійснюється в програмному середовищі, так називаній системі інформаційної підтримки життєвого циклу виробів, що робить необхідним освоєння відповідного програмного забезпечення.

1.1 Етапи життєвого циклу виробу

Процес створення й існування технічних об'єктів постійно еволюціонує. Від кустарних ремісників, що працювали поодинці без креслень і розрахунків, техніка прийшла до колективного автоматизованого виробництва, де частина операцій виконує сама техніка. На сьогоднішній день за час свого існування виріб проходить ряд етапів від ідеї до утилізації. Сукупність етапів або послідовність процесів, через які проходить виріб за час свого існування, називається життєвим циклом виробу. Основні етапи життєвого циклу об'єкта проектування представлені на рисунку 1.1.

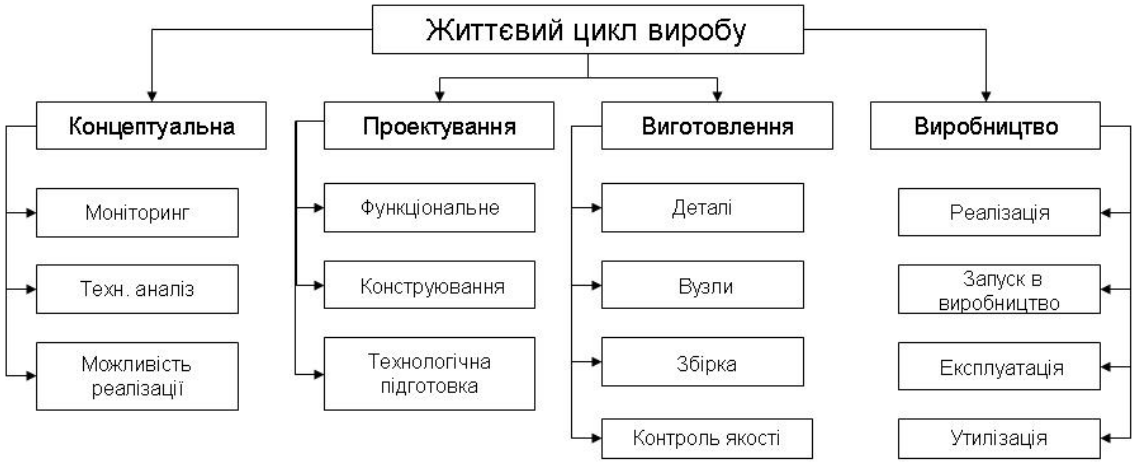


Рисунок 1 – Життєвий цикл об'єкта проектування

Життєвий цикл об'єкта проектування починається після маркетингових досліджень, які проводять виробники в пошуку споживачів своїх ідей або замовники в пошуку виконавців своїх завдань і потреб.

На першому етапі життєвого циклу, що іноді називають **концептуалізацією**, здійснюється технічний аналіз і формальне визначення потреб, а також оцінюється можливість фізичної реалізації виробу, що задовольнить потреби, буде при цьому конкурентоздатним й економічно вигідним. Замовник і виконавець формалізують свої потреби й ідеї у вигляді документа, що називається технічне завдання. Коли технічне завдання сформульоване і є впевненість, що воно буде повністю реалізовано, переходять до **проектування**.

У машинобудівних галузях промисловості прийнято виділяти системи **функціонального, конструкторського** й **технологічного** проектування. Перші з них називають системами розрахунків й інженерного аналізу або системами CAE (Computer Aided Engineering). Системи конструкторського проектування називають системами CAD (Computer Aided Design). Проектування технологічних процесів становить частину технологічної підготовки виробництва й виконується в системах CAM (Computer Aided Manufacturing).

Об'єктами **функціонального проектування** є схеми виробу. Саме тому функціональне проектування називають іноді схемним.

Схеми діляться по різних ознаках. Так, наприклад, залежно від фізичних принципів роботи тих або інших пристроїв розрізняють механічні, гідравлічні, пневматичні, електричні, кінематичні й інші схеми. Функціональне проектування є вкрай важливим етапом, з якого й починається власне проектування. Саме воно визначає оптимальність структури й характеристик функціональних пристроїв, блоків, вузлів й елементів виробу. Саме воно багато в чому забезпечує принципову можливість виконання виробу його головних завдань, гарантує одержання необхідних значень функціональних характеристик. Результатом функціонального проектування є різного роду схеми виробу і його частин. Проектування різних схем виконують інженери-проектувальники різних спеціальностей: механіки, електрики, електронники, автоматники.

Об'єктом **конструкторського проектування** (або просто конструювання) є просторова (твердотільна) структура виробу. На етапі конструювання спроектовані схеми перевтілюються реальні деталей і складальні одиниці, які розташовуються у просторі й закріплені певним чином. Говорять, що схеми при конструкторському проектуванні реалізуються "у залізо". Результатом конструкторського проектування є креслення (конструкторська документація). Їх розробляють, інженери-конструктори. Термін "конструктор" часто вживається й у більше широкому значенні, як синонім проектувальника взагалі.

Об'єктами **технологічного проектування** є технологічні процеси виготовлення деталей виробу. На цьому етапі розробляються технологічні документи: маршрутні карти (опис маршруту обробки деталей), операційні карти (опис виконуваних операцій), відомість оснащення (перелік використовуваних засобів технологічного оснащення) і ряд інших документів відповідно до єдиної системи конструкторської документації (ЕСКД)

(ДЕРЖСТАНДАРТ). Технологічне проектування виконують інженери-технологи різного профілю (механіки, електрики й інші)

Паралельно з технологічним проектуванням може здійснюватися виготовлення й випробування дослідних зразків окремих деталей і вузлів виробу або навіть усього виробу в цілому. При цьому проводяться виміри відповідальних деталей і різні випробування виробу. По їхніх результатах у проектну документацію вносять відповідні зміни. Це означає повернення на пройдені раніше етапи. Такі ж ітераційні повернення можуть здійснюватися як від технологічної до конструкторської галузі, так і від конструкторської до функціонального. Наприклад, це доводиться робити у випадку, коли розроблену конструкцію дуже складно й дорого (або взагалі неможливо) виготовити. У цих випадках говорять, що конструкція виявилася нетехнологічною. Буває, що розроблена схема конструюється погано або незручно, не поміщається в задані габарити й т.п. Про такі схемні рішення говорять, що вони неконструктивні. Таким чином, між етапами, як і розглянутими раніше рівнями, простежується явно ітераційний характер.

Виробництво виробів звичайно супроводжується специфічними процедурами виготовлення й контролю деталей або всього виробу в цілому. Важливим етапом є зборка виробу, у процесі якої потрібне моделювання роботи вже виготовлених деталей і вузлів машини, і проводити їхню оптимізацію.

Життєвий цикл виробу продовжують реалізація (продаж виробів кінцевим користувачам) і **експлуатація**, а завершує **утилізація**.

1.2 Основні відомості про автоматизоване проектування

Прогрес виробництва в сучасних умовах зв'язують із досягненнями в області автоматизації виробництва. Оскільки проектування й розробка технології є ступенем виробництва, то прогрес на цьому ступені також повинен визначатися автоматизацією.

При неавтоматизованому проектуванні результати багато в чому визначаються інженерною підготовкою конструкторів, їхнім виробничим досвідом, професійною інтуїцією й іншими факторами.

Автоматизоване проектування дозволяє:

1. Значно скоротити суб'єктивізм при прийнятті рішень,
2. Підвищити точність розрахунків,
3. Вибрати найкращі варіанти для реалізації на основі строгого математичного аналізу всіх або більшості варіантів проекту з оцінкою технічних, технологічних й економічних характеристик виробництва й експлуатації проектованого об'єкта.
4. Значно підвищити якість конструкторської документації,
5. Істотно скоротити строки проектування й передачі конструкторської документації у виробництво,

6. Ефективніше використати технологічне встаткування із програмним керуванням.

Автоматизація проектування сприяє більше повному використанню уніфікованих виробів і дозволяє поліпшити стандартні компоненти проектованого об'єкта.

З появою обчислювальної техніки нових поколінь й удосконалюванням методів її використання намітився новий, системний підхід до організації процесу проектування на ЕОМ, що полягає в створенні великих програмних комплексів у вигляді програмних модулів, пакетів програм (ПП) і САПР, орієнтованих на певний клас завдань. Такі комплекси будуються по модульному принципі з універсальними інформаційними й керуючими зв'язками між модулями, при рішенні завдань даного класу використаються єдині інформаційні масиви, організовані в банки даних.

Об'єднання декількох ПП у єдину систему, призначену для реалізації цілком певних функцій, дозволяє говорити про новий, більш високий рівень в ієрархії програмних комплексів. При цьому якісні зміни відбуваються і в організації інформаційного, технічного й іншого видів забезпечення, і, що особливо важливо, умови обміну інформацією між людиною й ЕОМ. Як правило, ці зміни спрямовані на підвищення гнучкості й універсальності системи, поліпшення характеристик взаємодії проектувальника з ЕОМ, підвищення якості одержуваного результату й зниження часу його одержання. Власне САПР можуть як підсистеми входити в системи більше високого рівня, наприклад АСУП (Рисунок 2).

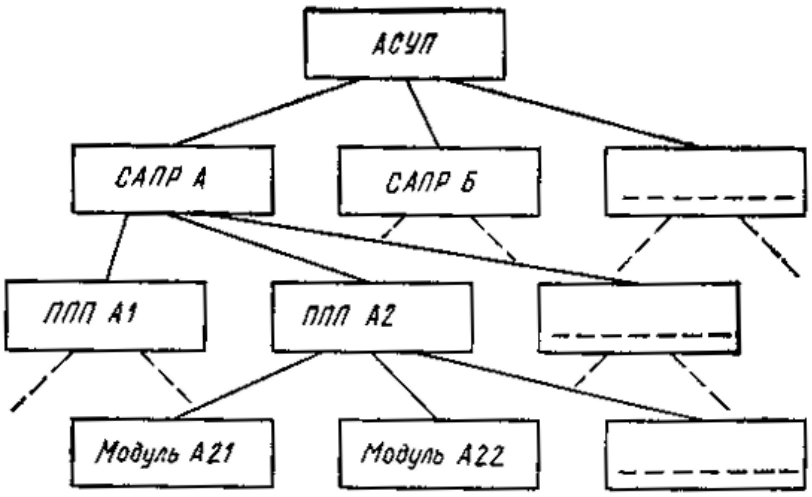


Рисунок 2 – Зв'язок модулів з системами

Формальне визначення систем проектування, що визначає її головні особливості – людино-машинна система, що використає сучасні математичні методи, засоби електронно-обчислювальної техніки й зв'язки, а також нові організаційні принципи проектування для знаходження й практичної реалізації найбільш ефективного проектного рішення відповідного об'єкта.

1.3 Етапи проектування в автоматизованих системах

Процес проектування можна розділити на наступні укрупнені етапи:

1. Пошук принципових проектних рішень.

2. Розробка ескізного варіанта конструкції і його оптимізація.

3. Уточнення й доробка обраного варіанта конструкції виконання повного детального розрахунку.

4. Розробка повного комплекту креслень и конструкторської документації.

Особливості цих етапів визначають ефективність застосування комп'ютера на кожному з них. На першому етапі значна роль евристичних дій. Повна автоматизація цього етапу можлива лише в деяких спеціальних випадках. Застосування комп'ютера тут найбільше доцільно й ефективно при організації діалогу між конструктором й комп'ютером, де конструкторові приділяються функції вибору й прийняття рішень, а комп'ютером — виконання дій по заданих алгоритмах, насамперед подання необхідної інформації і її обробка відповідно до завдання.

На другому етапі розглядаються різні конструктивні рішення з виконанням великої кількості операцій розрахунку й оптимізації, доцільне використання комп'ютерів шляхом створення систем діалогу із запрограмованим процесом конструювання й розрахункової оптимізації, при цьому конструктор направляє пошук оптимального варіанта конструкції й приймає рішення на підставі виконаних розрахунків.

Третій і четвертий етапи вимагають самих значних витрат часу й засобів (до 60%), причому розрахунково-конструкторська діяльність на цих етапах досить просто алгоритмізується, доцільне застосування на цих етапах комп'ютерів у комплексі із засобами уведення — виводу графічної інформації.

1.4 Вимоги та об'єкти проектування

Системи автоматизованого проектування повинні задовольняти наступним вимогам:

1. Автоматично виявляти найкращі проектні й проектно-технологічні рішення у всіх випадках, коли оптимізаційне завдання піддаються формалізації.

2. Автоматично вводити в процесі роботи системи інформації в усі взаємозалежні програми, що використають відповідну інформацію в якості вихідної.

3. Автоматично перевіряти сумісність рішень, прийнятих по різних частинах й елементам проектованого об'єкта, і здійснювати (коли це можливо) коректування несумісних рішень без участі проектувальника.

4. Автоматично видавати у звичній для проектувальника формі деякі проміжні результати.

5. Видавати будь-які проміжні результати по запиту проектувальника.

6. Надавати проектувальникові можливість вносити довільні корективи в рішення прийняті системою.

7. Давати можливість автономного рішення окремих задач по завданим проектувальником вихідним даним.

8. Накопичувати досвід проектування.

9. Видавати по запиту будь-які відомості, що зберігаються в банку дані системи.

10. Забезпечувати можливість удосконалювання й розвитку системи без її корінної переробки.

Всі ці вимоги можуть бути зведені до двох найважливіших якостей системи: інформативність й адекватність. Саме вони практично повністю визначають успіх впровадження та експлуатації.

Як правило, системи автоматизованого проектування призначені для проектування складних об'єктів.

Складним об'єктом проектування вважається виріб або споруда, що характеризується наступними ознаками:

1. складається з великої кількості елементів (деталей конструкції й комплектуючих виробів);
2. відрізняється суперечливістю вимог, пропонованих до його якості;
3. відрізняється не розробленістю формалізованих залежностей показників його якостей в залежності від прийнятих рішень або відсутністю однозначних критеріїв оцінки цих рішень;
4. має сукупність властивостей, обумовлених не тільки властивостями елементів, але й характером взаємодії між елементами;
5. відрізняється новизною технічних рішень;
6. призначається для експлуатації в складі багатокомпонентної системи або в мінливим (не цілком певним чином) умовах;
7. виготовляється із залученням великої кількості підприємств або з використанням індивідуальної технології.

1.5 Специфіка проектування обладнання в машинобудуванні.

Практика розробки й експлуатації систем проектування показує, що ряд особливостей побудови автоматизованої системи може бути обговорений і сформульований до початку її проектування. Специфіка об'єкта проектування накладає вимоги на структуру й організацію проектних робіт наприклад, млинів, пресових, валкових машин. Вона полягає в наступному:

1. Це «старий» об'єкт, розрахунками й проектуванням якого фахівці займаються вже більше 100 років, отже, всі принципові поліпшення конструкції вже, як правило, внесені. Тому прагнення до підвищення техніко-економічних показників змушує проектувальників усе глибше вникати в суть фізичних процесів, що досягається на шляху послідовного ускладнення математичної моделі й адекватного відображення процесів, що протікають у ньому, у тому числі й при перехідних режимах.

2. Якщо трудомісткість машини прийняти за 100%, то трудомісткість окремих проектних процедур розподілиться орієнтовно в такий спосіб:

– огляд існуючих конструкцій і визначення патентної чистоти виробу 2-3%;
– виконання розрахунків 6-14%;
– пророоблення конструкції 12-20%.;
– виконання креслень 37-55%;
– узгодження технічної документації 6-18%;
– оформлення технічної документації 9-16%.

1.6 Забезпечення систем проектування

Існуючий досвід в області автоматизації проектування свідчить про те, що розробка, впровадження й ефективне використання програмних комплексів, призначених для автоматизації процесу проектування й реалізованих на базі сучасних комп'ютерів, вимагають комплексного рішення широкого спектра проблем: організаційних технічних, математичних, програмних, лінгвістичних, інформаційних й ін. Рішення цих проблем базується на відповідних видах забезпечення.

Основними структурними елементами систем є підсистеми, які підрозділяються на ті що проектують й обслуговують. До тих що проектують відносять підсистеми, що виконують проектні процедури й операції, наприклад розрахункову, креслярсько-графічну, підсистему підготовки носіїв для верстатів із числовим програмним керуванням (ЧПУ) і автоматизованих ліній. До тих що обслуговують пристрої обчислювальної й організаційної техніки, засоби передачі даних, вимірювальні й інші пристрої або їхні поєднання.

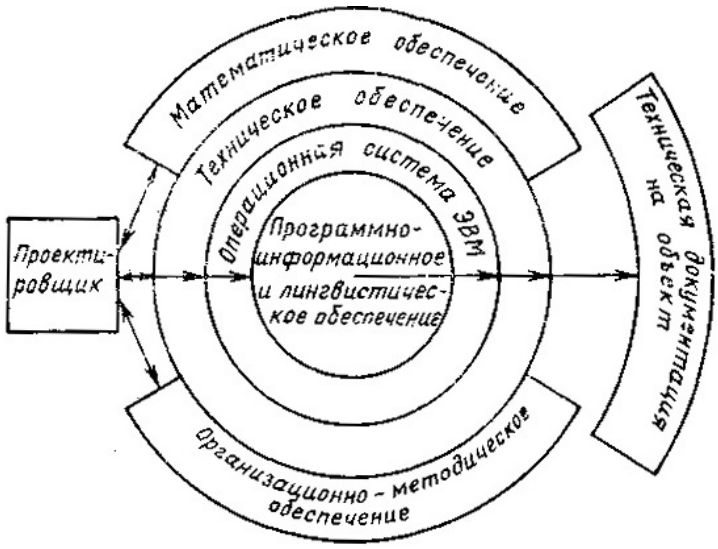


Рисунок 3 – Види забезпечення систем

Системи автоматизованого проектування містять наступні види забезпечення (Рисунок 3):

1. технічне – пристроїв обчислювальної й організаційної техніки, засобів передачі даних, вимірювальні й інші пристрої або їхні сполучення;

2. математичне – методи, моделі, алгоритми;

3. лінгвістичне – мови проектування, термінологія;

4. програмне – документи з текстами програм, програми на машинних носіях й експлуатаційні документи;

5. інформаційне – документи, що містять описи стандартних проектних процедур, типових проектних рішень, типових елементів, що комплектують виробів, матеріалів й інші дані, а також файли й блоки даних на машинних носіях із записом зазначених документів;

6. методичне – документи, у яких відбиті состав, правила відбору й експлуатації засобів автоматизації проектування;

7. організаційне – положення, інструкції, накази, штатні розклади, кваліфікаційні вимоги й інші документи, що регламентують організаційну структуру підрозділів й їхня взаємодія з комплексом засобів автоматизації проектування.

1.7 Пакети прикладних програм

Загальними особливостями є:

1. Орієнтація пакета на певний клас завдань. При цьому ПП діляться на *методо-орієнтвані* й *проблемно-орієнтовані.* Перші призначені для рішення задач різними методами, наприклад пакет алгоритмів параметричної оптимізації. Другі пакети призначені для рішення деякого набору задач, що розрізняються як по постановці, так і по методах рішення. Прикладом проблемно-орієнтованого ПП є пакет прикладних програм розрахунку оболонкових конструкцій.

2. Кожен ПП має деякий набір можливостей по методах обробки даних, формам подання результуючої інформації й т.д. Це дає можливість користувачеві вибрати необхідний варіант обробки даних.

3. Використання ПП передбачає суттєве зниження вимог до рівня професійної підготовки інженера-користувача в області програмування в порівнянні з підготовкою математика-програміста.

При класифікації ПП звичайно вказується тип операційної системи, під управлінням якої працює пакет, і спосіб організації пакета. По способу організації ПП діляться на пакети з *бібліотечною* й *блоковою* організацією. Перша є найбільше простою й орієнтована на користувачів-програмістів високої кваліфікації. При такому підході мало проблем при формуванні пакета, однак з'являються серйозні труднощі при експлуатації пакета, оскільки об’єднання програм за інформацією постійно вимагає втручання користувача на рівні алгоритмічної мови.

В основі побудови ПП завжди лежить *модульний* принцип (див. Рисунок 2). Модулем називається універсальний елемент ПП, що може застосовуватися самостійно або в пакеті програм. Модуль складається з тіла й паспорта. Тіло модуля - текст алгоритмічною мовою, написаний з дотриманням угод про склад за інформацією й керуванням між модулями. Паспорт модуля - сукупність найважливіших внутрішніх і зовнішніх характеристик модуля, тобто ідентифікатор модуля; мова програмування; списки вхідних, вихідних і проміжних параметрів;

вказівка на математичний апарат, що використовується у модулі. Список модулів і стандартних підпрограм, виклик яких здійснюється з тіла даного модуля; машинні ресурси, необхідні для роботи модуля (час функціонування, об’єм пам'яті й ін.).

Серед програмістів немає єдиної думки в питанні про розміри модуля. Очевидно для кожного конкретного випадку можливий оптимальний розмір, що повинно бути першоосновою для синтезу оптимальної структури системи.

1.8 Основні принципи створення систем проектування

Проблема синтезу оптимальної структури системи проектування вирішуються як шляхом використання досвіду створення автоматизованих систем керування, так і за рахунок нагромадження й використання досвіду створення й експлуатації великих програмних комплексів. Цей досвід концентрується в ряді принципів, на якими доцільно орієнтуватися при розробці [2]:

1. *Сумісність автоматичного й автоматизованого способів проектування.* Використання цього принципу дозволяє замінити один режим проектування іншим, більш продуктивним у кожному конкретному випадку, без зміни структури всієї системи в цілому.

2. *Автономність окремих частин системи,* побудованої на модульному принципі. Цей принцип припускає можливість незалежної розробки й незалежного запровадження в дію окремих частин системи, що дозволяє розширювати й ускладнювати систему в процесі її експлуатації.

3. Забезпечення *інтерактивного режиму проектування,* що дозволяє проектувальникові активно втручатися в цей процес, здійснювати контроль за ходом проектування в режимі діалогу «людина — машина». Організація інтерактивного проектування поліпшується зі зменшенням часу реакції ЕОМ.

4. *Мінімальність взаємодії системи із зовнішнім середовищем,* що припускає мінімізацію різних видів взаємодії системи із зовнішнім середовищем за рахунок скорочення обсягів вихідний й особливо вхідної інформації.

5. *Принцип розвитку, що* дозволяє робити модернізацію системи й розширення її можливостей за рахунок удосконалювання компонентів й упорядкованості зв'язків між цими компонентами без перерви або з мінімальними перервами у функціонуванні системи.

6. *Єдиний принцип побудови* систем автоматизованого проектування для групи родинних по функціональних характеристиках об'єктів.

7. *Принцип еволюційності* в проектуванні, тобто максимальне використання наявного досвіду й навичок проектування, перенесення їх у комплекс алгоритмів і програм, які є інструментом машинного проектування. Удосконалювання компонентів САПР у цьому випадку повинне базуватися на основі використання методів евристичного програмування.

8. *Принцип максимальної незалежності від технічних засобів,* які постійно обновляються (період відновлення поколінь комп’ютерів становить 2 — 3 роки).

9. *Принцип системної єдності, що* полягає в тому, що система будується як сукупність підсистем, функціонування яких спрямоване до загальної мети.

10. *Принцип наскрізного проектування, що* забезпечує безперервний характер проектування об'єкта від елемента до виробу в цілому й припускає автоматизацію на різних етапах проектування від задуму до втілення.

11. *Принцип ієрархічної побудови системи, що обумовлює* багатоступінчасту пірамідальну структуру системи з підпорядкуванням нижчих ланок вищим.

12. *Принцип включення, що* передбачає узгодження параметрів і можливостей конкретної системи *із* більше складною системою (АСУП, автоматизованою системою наукових досліджень АСНИ), що знаходиться вище на ієрархічному рівні.

13. *Принцип інформаційної єдності, що* вимагає використання у всіх підсистемах САПР нормативно встановлених у галузі правил застосування термінів, символів, способів подання інформації й т.д.

14. *Принцип моральної живучості, що* припускає наявність засобів настроювання на клас об'єктів, що розвиваються і змінюються як кількісно, так й якісно.

15. *Принцип першого керівника,* відповідно до якого за процеси розробки, впровадження й розвитку системи проектування повинен безпосередньо відповідати керівник відповідного проектно-конструкторського підрозділу. Спроби передоручити це керування другорядним особам, як правило, закінчуються тим, що система виявляється дискредитованою, або виконують рутинні функції.

16. *Принцип нових завдань* припускає рішення на базі системи проектування таких завдань, які без цих систем вирішувалися частково, приблизно або не вирішувалися взагалі через відсутність відповідних для цього ресурсів.

1.9 Особливості технології та вимоги до систем автоматизованого проектування

Проектування як форма інженерної діяльності має ряд характерних рис:

1. Процес має ітераційний характер.

2. Рішення приймаються на окремих етапах в умовах неповної або недостатньої інформації, що у цих випадках надходить або із зовнішнього середовища, або виробляється проектувальником у процесі творчої діяльності.

3. У процесі проектування поєднуються процедури алгоритмічного й евристичного характеру.

4. У проектній діяльності використаються різні ресурси, серед яких одним з найбільш важливих є знання проектувальника.

5. Ціль проектування встановлюється поза процесом проектування й залишається незмінною протягом цього процесу.

6. Процес проектування надає інформацію, що може бути використана у виробництві.

Системи автоматизованого проектування у процесі проектування відіграє роль потужного засобу, ефективне використання котрого неможливе без розробки комплексу методичних вказівок й інструкцій. Вони повинні регламентувати послідовність етапів й їхнє використання. Оскільки метою будь-якого процесу проектування є синтез конструктивного варіанта об'єкта, найбільшою мірою задовольняючим вимогам технічного завдання.

Реалізація технології автоматизованого проектування пред'являє комплекс наступних вимог:

1. можливість формулювати розв'язувані проектні завдання із предметної області на різних мовах, зрозумілих проектувальникові;
2. наявність засобів для ефективного коректування завдання на проектування з використанням простих форм вхідної мови (таблиць, бланків і т.п.);
3. відсутність жорстких обмежень на структуру й обсяг вхідних даних і форми носіїв інформації, на яких вони зберігаються;
4. можливість оперативного підключення до програмного забезпечення системи нових модулів і виключення застарілих;
5. подання можливостей проектувальникові на основі проміжних результатів ухвалювати рішення щодо виборі методів для продовження проектного завдання, а також змін значень окремих параметрів у використовуваному методі рішення;
6. можливість у ході виконання проектних операцій простежувати значення основних показників процесу, що свідчать про його ефективність, і залежно від їхніх значень коректувати обчислювальний процес;
7. допустимість включення навчальних програм для підвищення кваліфікації проектувальника;
8. забезпечення сумісності автоматизованого й неавтоматизованого видів проектування.

1.10 Проблеми функціонування людини і системи автоматизованого проектування

САПР створюється як людино-машинна система, у якій для автоматизації розумово-формальних процесів діяльності фахівців використаються можливості обчислювальної техніки. Процес проектування в системі організується таким чином, щоб підвищити віддачу розумової праці людини, стимулювати його творчу діяльність.

При цьому виникають деякі соціальні ефекти [17]. Зміст роботи таких людей радикально міняється, хоча й може мати різний характер, наприклад, якщо висококваліфікована робота, що вимагає знань і точності, але невеликої уяви, передається машині й виконується пакетом програм системи, то інженери, що постійно займаються такою роботою, стають операторами, що вводять дані в машину й отримують результати для аналізу. У розвинених діалогових системах, де проектні рішення приймаються людиною, а проміжна

обробка даних виконується машиною, ритм роботи радикально міняється. При цьому проектувальник нерідко витримує значні перевантаження, тому що темп роботи задається ЕОМ.

Обмеженнями в процесі використання систем автоматизованого проектування є припустимі навантаження елементів (обсяги інформації), що переробляють в одиницю часу. Найбільш слабким елементів системи є людина, що, як показують дослідження, нерідко змушена приймати рішення в процесі роботи в системі зі швидкістю в багато разів більшою, ніж при традиційній роботі. Природно, що людина не може тривалий час витримувати таке навантаження. Тому, наприклад, при роботі за дисплеєм проектувальник знижує свою продуктивність на 30-40% послу першої години роботи й на 70-80% послу другої години [3].

Так, за деяким даними частка творчої діяльності в загальному бюджеті часу проектувальника, що не використає автоматизовані системи проектування, становить усього 15%; формальна рутинна робота по праці становить 42%, формальна рутинна робота - 35%, інше - допоміжні й підготовчі роботи.

Таким чином, проектувальники досить високого рівня за допомогою систем автоматизованого проектування має змогу підвищити продуктивність праці і якість продукту. Однак робота при цьому стає більше напруженою, оскільки втрачаються періоди відновлення, які до впровадження систем визначалися виконанням рутинної роботи. Отже, із впровадженням систем проектування потреба в проектувальниках низької кваліфікації, коло обов'язків яких обмежувався саме рутинною роботою, буде зменшуватися.

1.11 Впровадження конструкторських систем на підприємствах.

Створенню системи автоматизованого проектування об'єкта, як правило, передує системне обстеження об'єкта проектування й використовуваних в інженерній практиці неавтоматизованих методів і прийомів проектування, технічної документації, розроблювальної в процесі проектування.

У результаті обстеження визначаються необхідність й економічна ефективність створення автоматизованої системи. При цьому враховуються обсяг проектно-конструкторських робіт, їхня періодичність, загальні витрати інженерної праці, можливість створення адекватного математичного опису й оптимізаційних процедур, необхідність підвищення техніко-економічних показників проектованого виробу й зменшення строків проектування й т. д

Істотною обставиною при рішенні питання про доцільність створення систем автоматизованого проектування є підготовленість відповідного проектного підрозділу (проектний інститут, конструкторське бюро й т.д.) до створення й впровадження систем.

Підготовленість до впровадження на підприємствах може бути оцінена по наступних ознаках:

- наявності математичних моделей об'єктів проектування й можливості їхньої реалізації на ЕОМ у процесі проектування;

- наявності технічних засобів (ЕОМ, периферійних пристроїв) і визначенню необхідності установки додаткових технічних засобів;
- готовності інформаційних фондів як у змісті їхньої впорядкованості з метою зберігання на машинних носіях, так й організованості технічних засобів зберігання й обробки інформації.

Крім оцінки підготовленості до розробки повинні бути виконані оцінки підготовленості до впровадження розробленої системи проектування. Ця оцінка містить у собі такі додаткові фактори:

- підготовленість кадрів проектно-конструкторської організації до роботи в умовах САПР;
- відповідність впроваджуваної системи прийнятої організації проектних робіт;
- психологічна підготовленість колективу до експлуатації системи і практичному використанню результатів автоматизованого проектування;
- відношення керівництва проектно-конструкторської організації до впровадження й використання системи.

Негативні оцінки по якому-небудь аспекті даної проблеми свідчать про необхідність проведення підготовчих робіт з розробки й впровадження систем автоматизованого проектування на даному підприємстві.

ТЕМА 2 СИСТЕМИ АВТОМАТИЗОВАНТГО ПРОЕКТУВАННЯ

Методи моделювання роботи конструкцій при дії на них різних фізичних навантажень, без натурного моделювання, виникли досить давно. Поява і розвиток обчислювальної техніки дало новий поштовх удосконалюванню чисельних методів аналізу, що є сьогодні основним інструментом конструктора. Системи автоматизації проектувальних робіт, які засновані на чисельних методах, стали невід'ємною частиною процесу конструювання деталей та виробів. Для успішного застосування кожен розрахунковий пакет повинний відповідати двом вимогам:

1. Утілювати найефективніші чисельні алгоритми;
2. Надавати користувачу розвинутій набір сервісних функцій по підготовці вихідних даних і обробці результатів розрахунку.

У залежності від ступеня відповідності даним критеріям усі програмні засоби автоматизації підрозділяються на легкі, середні і важкі. Ця ступінь, у даному випадку, є показником потужності й ефективності. Розглянемо можливості деяких комплексів орієнтованих на автоматизацію проектування, підготовку виробництва, інженерних розрахунків і аналізу конструкцій.

2.1 Системи для розрахунку на міцність

ANSYS - Скінченоелементный пакет. Найбільш використовуваний засіб для забезпечення інженерних розрахунків у світі. Універсальний розрахунковий комплекс, що застосовується у різних видах аналізу. Використовується для розрахунку конструкцій різного типу (авіабудування, суднобудування, машинобудування, будівництво, енергетика, електронна промисловість і ін.) з врахуванням різних фізичних впливів. З його допомогою виконуються як лінійні, так і нелінійні статичні та динамічні розрахунки конструкцій, аналіз усталостних руйнувань, рішення лінійних і нелінійних задач стійкості і теплофізики. Задачі гідро- і газодинаміки, акустики, електродинаміки й електростатики, п'єзоелектрика. Єдиний із представлених на світовому ринку комплекс, за допомогою якого, з використанням однієї бази, розв’язуються задачі типу теплофізика–міцність, електродинаміка–міцність, гідро–газодинаміка і міцність і ін. ANSYS дозволяє конструктору ще в процесі проектування передбачити поводження виробу і провести розрахунок на міцність, модальний і тепловий аналізи. Звести задачу до значень напружень, деформацій та розподілу температур і теплових потоків, які виникають при роботі виробу. Ґрунтуючись на виведених програмою колірних контурах, що являють градації «необхідності» матеріалу (залишити, забрати), конструктор може вилучити непотрібний матеріал, підводячи конструкцію до оптимального вагового співвідношення. Розроблювач – ANSYS Inc., США. http://www.ansys.com http://www.cosmos.rcnet.ru http://www.spark.nstu.nsk.su

ADAMS - Динаміка і кінематика складних механічних схем (механізмів) довільного виду, у т.ч. у реальному масштабі часу. Двосторонній зв'язок з

більшістю кінцево-елементних пакетів. Візуалізація результатів моделювання (відеореалістична анімація). Спеціалізовані модулі для конкретних галузей промисловості. Розроблювач - Mechanical Dynamics, Inc., США. http://www.designtechnologies.com

Pro/ENGINEER - Система високого рівня, САПР для єдиного циклу проектування–виробництво. Програмний комплекс Pro/ENGINEER охоплює весь цикл http://www.ptc.com/«конструювання – виробництво» у машинобудуванні. В усьому світі більш 16000 компаній використовують програмні продукти фірми PTC для скорочення тривалості наскрізних проектно-виробничих циклів, оптимізації інженерних процесів і поліпшення якості продукції. Ядро Pro/ENGINEER використовує унікальну по своїх можливостях технологію - Proven Technology, засновану на граничних представленнях. Розроблювач - Parametric Technology Corporation, США. http://www.ptc.com http://www.cv.com http://www.solver-net.com http://www.solver-net.ru

CATIA/CADAM Solutions - CATIA (Computer Aided Three-dimensional Interactive Application) - це цілком інтегрована універсальна CAD/CAM/CAE система http://www.catia.com/високого рівня, що дозволяє забезпечити рівнобіжне проведення конструкторсько-виробничого циклу CATIA. Будучи універсальною системою автоматизованого http://www.3ds.com/проектування, , широко застосовується на великих машинобудівних підприємствах в усьому світі для автоматизованого проектування, підготовки виробництва, реінженерінга, випробування і виготовлення деталей та вузлів. Число фірм-користувачів CATIA перевищує 8 тисяч. Функції, підтримувані CATIA/CADAM Solutions: · адміністрування - планування, керування ресурсами, інспектування і документування проекту;· опис усіх механічних зв'язків між компонентами об'єкта і приведення їх у стан просторового взаємного позиціювання;· автоматичний аналіз геометричних і логічних конфліктів· аналіз властивостей складних складальних креслень;· розроблений інструментарій трасувань систем комунікацій з дотриманням заданих обмежень;· спеціалізовані додатки для технологічної підготовки виробництва. Компанії DASSAULT SYSTEMES (Франція) і IBM (США) є спільними розроблювачами і розповсюджувачами цієї системи автоматизованого проектування. http://www.catia.ibm.com http://www.catia.com http://www.catia.ru http://www.ibm.ru http://www.catia.spb.ru

LS-DYNA - багатоцільова програма , що використовує метод кінцевих елементів (МКЕ), - призначена для аналізу нелінійного динамічного відгуку тривимірних пружних структур. Включає високоефективний алгоритм рішення нелінійних і швидкоминучих процесів, автоматизований процес рішення контактних задач, а також безліч функцій по перевірці одержуваного рішення дозволяють інженерам в усім світі успішно вирішувати важкі завдання удару, руйнування й формування. Унікальний математичний апарат включає більше 25 алгоритмів контактної взаємодії,

більше 100 рівнянь стану, що дозволяє вирішувати завдання: Нелінійної динаміки, Теплові; Руйнування; Розвитку тріщин; Контакту; Квазистатики; Ейлерового формулювання МКЕ; Довільного лагранж-ейлерова поводження; Акустики в реальному масштабі часу; Багатодисциплінарного аналізу: міцність, теплофізика, акустика;

Eta/DYNAFORM - спеціалізований програмний комплекс, орієнтований на моделювання процесів листового штампування й математичний апарат, що використає як ядро, програми LS-DYNA. Пре- і постпроцессинг DYNAFORM побудований з урахуванням всіх специфічних особливостей техпроцесса: він автоматизує стандартні операції підготовки розрахункової схеми й функції оцінки й інтерпретації результатів аналізу й базується на загальноприйнятій термінології, знайомій кожному інженерові-технологові. Інструментарій програми включає: Автоматична побудова сіток; Адаптивні сітки з анімацією історії побудови; Велику бібліотеку промислових матеріалів; Автоматизоване позиціювання інструмента; Залучення явищ втрати стійкості аркуша - жолоблення; Розрахунок тангенціальних зусиль під притисками (гальмовими ребрами); Розрахунок пружного розвантаження виробу ; Високоякісну візуалізацію всіх результатів й анімацію; Побудова граничної діаграми «формуемости».

CADfix включає унікальний набір засобів по відновленню геометричних моделей, аж до одержання твердотільної моделі по незв'язаному каркасному наборі опорних ліній, а також по модифікації й експорту геометричних файлів. Одним з найважливіших призначень CADfix – створення розрахункових моделей для кінцево-елементного розрахунку - доведення твердотільної геометрії до прийнятного для розбивки стану й безпосередня розбивка на кінцеві елементи. Можливості CADfix: Автоматичне сканування й візуалізація виявлених проблем з підказкою методики вирішення; Автоматизована ітераційна процедура відновлення моделі; Обрізка поверхонь; Зшивка в межах автоматично обумовленої або точності, що задає користувачем; Розбивка твердих тіл на більше прості складові; Наявність власного сіткового генератора;

C-MOLD виконує чисельне моделювання процесів обробки пластмас. Заснована на МКЕ програма C-MOLD призначена для комп'ютерного моделювання процесів обробки всіх видів пластмас. У програмі реалізоване моделювання велика кількість технологічних процесів, зокрема: лиття, термопластів під тиском, інжекційного лиття із застосуванням газу, процесів двохкомпонентного лиття, пневмовакуумформування з урахуванням явищ усадки й жолоблення й багатьох інших, а крім того, розрахунок параметрів матеріалу й виробу на всіх стадіях обробки з можливістю оптимізації як форми лиття (положення, форма літників і т.д.), так і самого виробу.

2.2 Системи для розв’язку задач аерогідродинаміки

Star-CD була першої у світі програмою, що включила в себе процедуру так званих ковзних сіток. Ефективна паралелизація алгоритму рішення, заснованого на застосуванні методу кінцевих об'ємів, у сполученні з унікальними методиками автоматизованої розбивки області течії дозволяє моделювати завдання будь-якого ступеня геометричної складності. Традиційними галузями застосування Star-CD є наступні: Транспорт; Енергетика; Хімічна й обробна; Загальне машинобудування; Будівельна; Електротехнічна й електронна; Газо- і нафтовидобуток;

Star-CD є багатоцільовим єдиним CFD-пакетом, що надає користувачеві наступні можливості для рішення завдань механіки рідин і газів на всіх типах сіток: Стаціонарні й нестаціонарні течії; Ламінарні течії - модель Ньютона й неньютонівські рідини; Турбулентні течії (застосовується трохи найбільш відомих моделей); Стисливі, нестисливі (включаючи біля- і надзвукові); Теплопереніс (конвективний, радіаційний, теплопровідність із урахуванням твердих тіл); Массопереніс; Хімічні реакції; Горіння газоподібного, рідкого й твердого палива; Розподілений опір (наприклад, у пористих середовищах, теплообмінниках); Багатокомпонентні плини. www.cadfem.ru/program/star_cd/star_cd.htm

CFX - програмний комплекс, призначений для розрахунку задач аэро- і гідродинаміки. З 2003 року CFX входить до складу програмного комплексу ANSYS, більше правильна назва ANSYS CFX. Сполучає унікальні можливості аналізу гідрогазодинамічних процесів, багатофазних потоків, хімічної кінетики, горіння, радіаційного теплообміну й багатьох інших. CFX забезпечує принципово новий рівень рішення завдань обчислювальної гідрогазодинаміки за рахунок унікального сполучення технологій, починаючи від прямого інтерфейсу до більшості CAD систем і закінчуючи можливістю проводити сполучений рідинно-структурний аналіз разом з ANSYS Multiphysics. Широкий вибір моделей турбулентності, у сполученні з лінійним решателем з технологією "Algebraic Coupled Multigrid" дозволяє домогтися високої точності результатів при рішенні різного класу завдань. CFX є визнаним лідером при розрахунках турбомашин. Програма CFX має широкий спектр додатків, по основних галузях: аерокосмічна, автомобілебудування, суднобудування й морська техніка, нафтогазова й хімічна, турбомашинобудування, теплотехніка, вентиляція й кондиціонування, біомедицинські додатки.

FLUENT - програмний комплекс, призначений для рішення завдань механіки рідин і газів. З 2006 року FLUENT входить до складу програмного комплексу ANSYS. Використає неструктуровану сіткову технологію (типи елементів - гексаэдры, тетраедри, призми й піраміди). Адаптація розрахункової сітки дозволяє отримати точне рішення для областей з великими градієнтами потоку, наприклад, для прикордонних шарів. Можливості використати моделі динамічної адаптивної сітки дозволяють проводити наступні

розрахунки: потоки в циліндрах, клапани й інших. FLUENT дозволяє проводити спільні розрахунки ротор-статор для турбомашин, використовуючи технологію ковзних сіток. В FLUENT включені ламінарні й турбулентні моделі гідродинаміки, теплопередачі, фазових переходів і радіації, а також моделі для розрахунку кавітації, стисливих рідин, теплообміну, теплопровідності, реальних газів, і модуль для розрахунку вологої пари. FlUENT є визнаним лідером при розрахунку хімічних реакцій і завдань горіння. FLUENT використає найсучасніші підходи для моделювання хімічних реакцій. База даних містить властивості багатьох газоподібних, твердих, пиловугільних, і рідких паливних матеріалів. Також доступні моделі для пророкування NO_x. FLUENT містить у собі безліч моделей турбулентності: кілька версій моделі /k/-epsilon, моделі /k/-omega, Reynolds stress модель (RSM), LES модель, DES модель. Один з кращих пакетів в області багатофазного моделювання. Різні можливості програми дозволяють отримувати найглибші відомості про встаткування. FLUENT використає такі моделі як: VOF, mixture і модель Эйлера. Для деяких багатофазних явищ може використатися модель дискретної фази (DPM). VOF моделі використаються для розрахунку течій з вільними поверхнями. Модель кавітації дозволяє розрахувати насоси, паливні інжектори, корабельні гвинти. FLUENT дозволяє виконувати паралельні обчислення на Windows, Linux, і Unix платформах. При цьому можуть бути використані многопроцессорные машини або кластери. FLUENT має клієнт-серверну архітектуру. FLUENT характеризується досить гарною масштабованістю. www.processflow.ru/fluent.html, http://www.fluent.com/software/fluent/

FlowVision моделює тривимірні стаціонарні/нестаціонарні нестисливі потоки рідини в різних технічних додатках. Використання різних моделей турбулентності й адаптивної розрахункової сітки дозволяє моделювати складні рухи рідини, включаючи плини із сильним закрученням і з горінням. FlowVision заснований на кінцево-об'ємному методі рішення рівнянь гідродинаміки й використає прямокутну адаптивну сітку з локальним здрібнюванням. Для апроксимації криволінійної геометрії з підвищеною точністю FlowVision використає нову технологію - підсіточну геометрію. Ця технологія дозволяє імпортувати геометрію із САПР систем й обмінюватися інформацією із системами кінцево-елементного аналізу. FlowVision ефективно використається для рішення наступних промислових завдань: Зовнішня аэро- гідродинаміка: обтікання автомобіля, судна, літака, ракети, будинків і споруджень (визначення коефіцієнтів опору й піднімальної сили, розподілене навантаження, тепло- і массопереніс) Внутрішня аэро-гідродинаміка: течію у салоні автомобіля й у подкапотнім просторі, вентиляція внутрішніх відсіків, рух газів і рідин по магістралях і трубопроводам. Моделювання турбомашин: плин рідини у турбінах, компресорах, насосах, урахування впливу гребних гвинтів на обтікання судна. Моделювання процесів горіння; технологічних процесів: моделювання

теплопереносу в мікроелектронних схемах, розрахунок витрат-напірних характеристик ежекторного насоса, розрахунок змішувачів і газових міксерів, спільний теплоперенос між рідиною й твердими тілами. Моделювання процесів виготовлення деталей: лиття металів, розрахунок процесів затвердіння й кристалізації. www.flowvision.ru/ www.tesis.com.ru/1level/2level/cadme/programs/flowvis/long.html.

FLOW-3D - це CFD пакет загального призначення здатний моделювати велику розмаїтість завдань течії рідини й/або газу. Спеціалізацією пакета є моделювання течій з вільною поверхнею, FLOW-3D є чудовою програмою для моделювання обмежених і внутрішніх течій. FLOW-3D забезпечує високоточне моделювання течій з вільною поверхнею, використовуючи метод кінцевих об'ємів. Використовуваний в FLOW-3D метод FAVOR є унікальним методом, відсутнім в інших CFD пакетах. FAVOR досить простий у використанні й забезпечує високу точність моделювання. www.cad.ru/ru/software/detail.php?ID=3749, www.flow3d.com/Parallel.htm

GDTGasDynamicsTool - це програмний продукт, розроблений для обчислювального моделювання газо-динамічних процесів у широкому колі граничних і початкових умов. Текуче середовище, внутрішні й зовнішні проблеми аеродинаміки, балістика, горіння й детонація - це всього лише кілька явищ, які можна моделювати за допомогою GDT продукту. Сфери застосування: Екологія й захист навколишнього середовища, Машинобудівне проектування, Проектування просторових систем, Вивчення фізично - хімічних процесів горіння, Технології озброєння. Пакет GDT дозволяє: Зробити ефективним використання можливостей багатопроцесорних обчислювальних систем, Установлювати різні граничні умови в будь-якій точці області, Змінювати конфігурацію моделируемой проблеми що моделюється оперативно в ході обчислювального . Зчитувати значення кожного параметра в будь-якій крапці рахункової області в ході обчислення. Зберігати конфігурацію області й результати обчислень для наступного використання. Установлювати динамічні сценарії, що складаються з необмеженого числа кроків, у яких користувач може встановлювати різні початкові й граничні умови на кожному кроці. www.cfd.ru/

3D QuickFill - Програма, що дозволяє на ранніх стадіях проектування виробу провести аналіз лиття по тривимірної твердотільній моделі. Надає можливість конструктору спостерігати процес заповнення литєвої форми з наданням інформації про: часове заповнення прес-форми; час охолодження деталі; розподіл температури; наявність «раковин»; маса виробу. Розроблювач – Moldflow Corporation Advanced CAE Technologies, Inc., США. http://www.cmold.com http://www.autograph.ru/cad/3d_quickfill.htm

2.3 Системи проектування

AutoCAD - розробка компанії Autodesk - найпоширеніша САПР, що функціонує на платформі MS Windows. AutoCAD — це традиційний, перевірений часом інструмент інженерної графіки, тривимірного моделювання і візуалізації, що постійно доповнюється новими можливостями. Платформа AutoCAD забезпечує ефективне підвищення продуктивності праці в будь-якій області діяльності, зв'язаної з точним графічним представленням результатів — від астрономічних спостережень до розкрою одягу. Продукти AutoCAD призначені для виконання широкого спектру інженерних робіт в таких областях як будівництво та архітектура, картографія, геодезія, машинобудування. Функціонал AutoCAD доповнюють більш як 5000 спеціалізованих програм–додатків для найрізноманітніших галузей. До них відносяться AutoCAD, Internet Publishing Kit, AutoCAD LT, S8 Architectural, S8 Architectural Designer, S8 Architectural Professional, S8 Building Services, S8 Civil/Survey, S8 Survey Professional, AutoCAD AEC (UK/Ire.), AutoCAD Mechanical, Autodesk Mechanical Desktop, AutoCAD Map та багато інших. Продукт Autodesk World дозволяє оперувати данними в середовищі Windows і Microsoft Office включаючи векторні та растрові зображення. http://usa.autodesk.com/

Genius - Продукти Genius є програмним забезпеченням для конструювання в машинобудуванні і створення креслень із застосуванням AutoCAD. Genius Desktop – Обєктно-орієнтована система тривимірного проектування машинобудівних деталей і складальних креслень на базі Mechanical Desktop. Пакет пропонує додаткові зручні інструменти для нанесення типових конструктивних елементів, наповнення конструкції стандартними виробами у виді твердотельних моделей, що значно полегшує роботу конструктора при управлінні компонентами складальних креслень. Genius має у своєму розпорядженні бібліотеки стандартних деталей у вигляді готових параметричних деталей по цілому рядіу типових стандартів.

КОМПАС - Один з лідируючих російських продуктів. CAD–система, призначена для широкого http://www.ascon.ru/спектра проектно-конструкторських робіт, легка в освоєнні, зручна в роботі і при цьому має не велику вартість, що є прийнятним для комплексного оснащення російських, українських та інших підприємств на території СНД. Дозволяє здійснювати двовимірне проектування і конструювання, швидку підготовку і випуск різноманітної креслярсько-конструкторської документації, створення технічних текстово-графічних документів. Розроблювач – Аскон, Росія. http://www.ascon.ru

CADMECH - CADMECH – система проектування деталей і складальних одиниць на базі AutoCAD. CADMECH http://www.intermech.ru/Desktop - тривимірна система проектування деталей і складальних одиниць на базі Mechanical Desktop. Розроблювач - НПО «Интермех», Мінськ. http://www.intermech.ru

Autodesk Inventor - Autodesk Inventor - програма тривимірного параметричного моделювання складних об'єктів. Робота зі складальними кресленнями налічує близько 13 000 компонентів. Прогресивний інтерфейс, що дозволяє освоїти

роботу з програмою за декілька днів, та підтримка даних, виконаних у програмах на платформі AutoCAD дозволяє в короткі строки будувати складні 3D моделі. http://www.inventor.ru/main.cfm

DYNAMIC DESIGNER - Інтегрованаий у середовище Machanical Desktop розрахунковий модуль для проведення динамічного і кінематичного аналізу механізмів. Розроблювач - Mechanical Dynamics, США.

DesignSpace - Програма, призначена для роботи в середовищі Mechanical Desktop, що вбудовується в меню Mechanical Desktop і використовує той же інтерфейс і геометрію. Пакет DesignSpace дозволяє проводити перевірочні і проектувальні розрахунки елементів http://www.mscsoftware.com/products/твердотільних конструкцій, створених у середовищі Autodesk Mechanical Desktop. В ньому реалізовані розрахунки статичної міцності конструкцій і визначення форм і частот власних коливань. Від конструктора не потрібно додаткових знань про розрахунок конструкції, йому потрібно мати тільки представлення про роботу своєї деталі в умовах реальної експлуатації. Від конструктора потрібно введеня тільки характеристик матеріалу деталі і зовнішніх факторів: прискорення, робочої температури, граничних умов і зовнішнього навантаження. Після розрахунку конструктору потрібно тільки відобразити результати і проаналізувати їх. Розроблювач – ANSYS Inc., США. http://www.designspace.com http://www.ansys.com http://www.cosmos.rcnet.ru http://www.spark.nstu.nsk.su

MSC/InCheck – інтегрований у Mechanical Desktop розрахунковий модуль для проведення http://www.mscsoftware.com/products/аналізу на міцність та оптимізацію конструкції: статична міцність (напруги і деформації); вібрації; стійкість; тепловий аналіз (теплопровідність і теплообмін). Розроблювач - The MacNeal-Schwendler Corporation, США. http://www.mscsoftware.co

T-FLEX CAD - Система параметричного проектування і креслення T-FLEX CAD є розробкою російської фірми «Топ Системи». Система забезпечує: параметричне проектування і моделювання; проектування і виконання складальних креслень; повний набір функцій створення і редагування креслень. Просторове моделювання, що базується на технології ACIS; параметричне тривимірне твердотільне моделювання; керування кресленнями; підготовка даних для систем із ЧПУ; імітація руху конструкції. Система T-FLEX CAD потрапила в огляд кращих САПР за 1997 рік. Розроблювач – Топ-Системи, Москва. http://www.tflex.com http://www.topsystems.ru

bCAD – програмний проект, спрямований на розробку новітніх технологій 3D графики і САПР, а також програм для 2D проектування та точне креслення, 3D моделювання і фотореалістичне тонування. Пердбаченаа програмна система 3D моделювання і візуалізації для PC. Система bCAD спроектована та розроблена як універсальне робоче місце проектувальника, що дозволяє робити широкий спектр робіт у наскрізному режимі – від креслення до об'ємної моделі і навпаки - від тривимірного представлення до плоских проекцій. Використовується для виконання технічної документації, що відповідають вимогам стандартів, та озробки реалістичних зображень при підготовці даних для розрахункових

систем. Система поєднує в собі CAD, 3D моделювання і фотореалістичну візуалізацію. Розроблювач - ProPro Group, Новосибірськ. http://www.propro.ru

CADRA - Система двовимірного проектування і креслення для машинобудування. Розроблювач – SofTECH, Inc., США. http://www.softech.com

CADkey - 3D графічний пакет для проектування, твердотільного, поверхневого і каркасного моделювання, візуалізації і документування простих і складних деталей і складальних одиниць. 250000 інсталяцій у різних країнах. Розроблювач - Baystate Technologies, США. http://www.cadkey.com http://www.cadkey.de http://www.colla.lv

DesignCAD Pro - Система двовимірного і тривимірного проектування і моделювання для професійних конструкторів і проектувальників. Розроблювач – ViaGrafix, США. http://www.viagrafix.com

IronCAD - Система автоматизованого проектування для машинобудування. Забезпечує двовимірне проектування і тривимірне твердотільне моделювання. Розроблювач - Visionary Design Systems, Inc., США. http://www.ironcad.com

Surface Express - Система поверхневого моделювання. Розроблювач – MCS, Inc., США. http://www.mcsaz.com

2.4 Комбіновані системи

SolidWorks - http://www.solidworkc.com/Потужний машинобудівний CAD пакет для твердотільного параметричного моделювання складних деталей та складальних креслень. Система конструювання середнього класу, що базується на параметричному геометричному ядрі Parasolid.Створена спеціально для використання на персональних комп'ютерах під керуванням операційних систем Windowsi. Розроблювач - SolidWorks Corporation, США. http://www.solidworks.com, http://www.uscad.com, http://www.delcam.ru, http://www.ascon.ru http://www.colla.lv

SolidEdge - http://www.solid-edge.com/SolidEdge є принципово новою системою автоматизованого конструювання, що призначена для розробки складальних вузлів і http://www.solid-edge.com/геометричного моделювання окремих деталей. Solid Edge розроблений спеціально для конструювання виробів машинобудування. Являє собою систему середнього рівня, що забезпечує ефективне обєктно-орієнтоване параметричне моделювання в середовищі MS Windows. Базується на ядрі геометричного моделювання Parasolid. Розроблювач – Unigraphics Solutions, США. http://www.solid-edge.com/ http://www.cosmos.rcnet.ru http://www.ugsolutions.ru http://www.solid-edge.com

Unigraphics-Система Unigraphics є CAD/CAM/CAE - системою високого рівня. Unigraphics дозволяє здійснювати цілком віртуальне проектування виробів, механообробка деталей http://www.plmsolutions-eds.com/products/unigraphics/складних форм, має цілком асоціативну базу даних майстрів-моделі, Unigraphics Solutions, одна із компаній, що швидко розвиваються, виробляючи системи http://www.eds.com/products/plm/unigraphics/ автоматизованого проектування, виробництва і керування проектами. Займається розробкою, продажем і

технічною підтримкою програмного забезпечення для автоматизації проектування, виробництва, інженерного аналізу і керування проектами для всіх областей промисловості, включаючи автомобілебудування, авіаційну і космічну промисловості, верстатобудування, виробництво товарів народного споживання і т.п. Серія продуктів Unigraphics Solutions, Inc.: Unigraphics Solutions, Parasolid, Solid Edge, Unigraphics, IMAN, ProductVision, GRIP. Розроблювач - Unigraphics Solutions, Inc., США. http://www.plmsolutions-eds.com/ http://www.cosmos.rcnet.ru/ http://www.ugsolutions.ru/ http://www.int.kiev.ua

CADdy - http://www.caddy.de/Система по функціональних можливостях займає проміжне положення між системами низького і високого рівнів. Призначена для рішення комплексних інтегрованих технологій від стадії проектування до стадії виробництва в таких областях, як:· архітектура;· проектування промислових установок;· машинобудування;· електроніка;· устаткування (опалення, вентиляція, сантехніка, електротехніка);· інженерні мережі і дороги;· геодезія, картографія. Розроблювач - фірма ZIEGLER-Informatics Gmb, Німеччина. http://www.caddy.de http://www.plaza.ch http://www.caddy.ru

OmniCAD - Система двовимірного проектування, креслення і тривимірного поверхневого моделювання. Розроблювач – CAMM s.r.l., Італія. http://www.camm.it

think3-Система автоматизованого проектування для машинобудування середнього рівня. Забезпечує http://www.think3.com/двовимірне проектування, тривимірне поверхневе і твердотільне моделювання, проектування виробів з листових матеріалів, ассоциативність двовимірного креслення з тривимірною моделлю, фотореалістичне представлення проекту. Розроблювач – thnk3, Inc., США. http://www.think3.com http://www.dial-eng.spb.ru

Cimatron – інтегрована CAD/CAM що надає повний набір засобів для конструювания виробів, розробки креслярсько–конструкторської документації, інженерного аналізу, створення керуючих програм для верстатів із ЧПУ.

TEBIS - CAD/CAM система. Двовимірне проектування та тривимірне моделювання в креслення. Розроблювач – Tebis AG, Німеччина. http://www.tebis.de

VISI–Series - CAD/CAM – система. З її допомогою забезпечується двовимірне проектування та тривимірне поверхневе і твердотільне моделювання, генерація програм для верстатів із ЧПУ, візуалізація обробки деталей. Розроблювач – Vero International, Inc., США. http://www.veroint.com

HELIX - HELIX Design System - система САПР для двовимірного та тривимірного проектування в машинобудуванні, дизайн в інших галузях. Дозволяє здійснювати двовимірне проектування та тривимірне каркасне, поверхневе і твердотільне моделювання. Розроблювач – MicroCADAM Ltd., Великобританія. http://www.cadam.com/ccd_microcadam.htm

Form-Z - Система двовимірного проектування в кресленні, тривимірного поверхневого і http://www.formz.com/твердотільного моделювання, візуалізації й

анімації для професійного дизайну, візуалізації в проектуванні. Розроблювач – Autodessys, Inc., США. http://www.formz.com

Alias/Wavefront - Розповсюджені програмні продукти двовимірного і тривимірного ескізування та креслення, тривимірного поверхневого і твердотільного моделювання, візуалізації й анімації, для професійного дизайну і проектування. Розроблювач - Alias/Wavefront, Канада. http://www.aw.sgi.com http://aliaswavwfront.com

CoCreate - Серія продуктів для проектування і управління даними проекту:
ME10 – http://www.cocreate.com/проектування і креслення;
SolidDesigner – твердотільне моделювання і управління даними проекту. Розроблювач – CoCreate Software, Inc., http://www.cocreate.com

CADMAX - CADMAX SolidMaster – система автоматизованого проектування, що забезпечує двовимірне, тривимірне поверхневе і твердотільне моделювання в проектуванні. Розроблювач – CADMAX Corp., США. http://www.cadmax.com

BRAVO – програмний комплекс для проектування, підготовки конструкторської документації, підготовки виробництва і управління проектом у машинобудуванні. Продукти: Bravo XL, Bravo Sheet Metal Fabricator, Bravo NCG, Bravo Frame. Розроблювач – Applicon, Inc., США. http://www.applicon.com

MicroStation - професійна, високо продуктивна система для 2D/3D - автоматизованого проектування при виконанні робіт зв'язаних з кресленням, конструюванням, візуалізацією, аналізом, керуванням базами даних і моделюванням. Забезпечує практично необмеженими можливостями проектувальників і конструкторів на платформах DOS, Windows і комп'ютерах різних типів. MicroStation 95 - система колективної роботи, що дає всім учасникам групи гарантію взаємного узгодження незалежно від апаратного розвитку платформ. Розроблювач – Bentley, США. http://www.bentley.com http://www.cosmos.rcnet.ru

hyperMILL-Пакетдля реалізації реалізультатів завершуючої технологічної ланки в http://www.openmindtech.com/CAD/CAM/CAE-технології для підготовки керуючих програм верстатів із ЧПУ і виготовлення виробів. Розроблювач - Open Mind Software Technologies Gmb, Німеччина. http://www.openmindtech.com http://www.acad.co.uk http://www.autodesk.com

EdgeCAM - CAM – система. Пропонує рішення для фрезерної, поверхневої, токарської й електроерозійної обробки деталей. Розроблювач – Pathtrace, Великобританія. http://www.pathtrace.com

ESPRIT - CAD/CAM – система на базі ядра Parasolid. Розроблювач – DP Technology, США. http://www.dptechnology.com

SolidCAM - Пакет генерації керуючих програм для верстатів із ЧПУ при обробці деталей, що http://www.solidcam.com/містять складну поверхневу чи твердотільну геометрію. Забезпечує 2,5 і 3-осьову фрезерну обробку, токарську обробку, візуалізацію процесу обробки. Розроблювач – CADTECH, Ізраїль. http://www.solidcam.com

MasterCAM - CAD/CAM – система, що займає лідируюче положення у світі по кількості продажів і http://www.mastercam.com/інсталяцій пакета серед CAD/CAM систем. Забезпечує каркасне і поверхневе моделювання деталей, візуалізацію і документування простих і складних деталей та складальних одиниць, розробку керуючих програм для токарської, фрезерної, електроерозійної обробки на верстатах із ЧПУ. Розроблювач - CNC Software, США. http://www.mastercam.com http://www.colla.lv

PEPS - CAM – система для автоматизованої підготовки фрезерної, токарської, лазерної, електроерозійної обробки деталей. Розроблювач – Camtek Ltd., Великобританія. http://www.camtek.co.uk

EUCLID - САПР високого рівня EUCLID, що охоплює всі етапи проектування. Передбачена розробка, продаж та супровід програмного забезпечення CAD/CAM/CAE/PDM і програмного середовища для http://www.matra-datavision.com/pages/subscription.htmlстворення додатків. Основні продукти фірми мають торгові марки: EUCLID, STRIM, PRELUDE, CAS.CADE і призначені для таких областей проектування як авіація, космос, автомобілебудування, оборона, електромеханіка, промисловий дизайн, атомне машинобудування, інжиніринг, виробництво товарів широкого вжитку й ін. Розроблювач - MATRA DATAVISION, Франція. http://www.matra-datavision.fr http://www.datavision.com http://www.eads.matradatavision.co.uk/ http://www.us.matradatavision.com/

I-DEAS Master Series - Програмний комплекс I-DEAS Master Series дає можливість оптимізувати концепцію виробу на ранній стадії проектування, що дозволяє значно поліпшити його якість при зменшенні часу розробки і витрат для розробки механічних виробів. Розроблювач - Structural Dynamics Research Corporation (SDRC), США. http://www.eds.com/products/plm/ideas/ http://www-win.scan.ru/scan/partners/SDRC

DEFCAR CAD/CAM – система для проектування і підготовки виробництва в кораблебудуванні. Розроблювач – Defcar Ingenieros, S.L., Іспанія. http://www.defcar.com

SoftCAD - САПР для двовимірного і тривимірного проектування в архітектурі і будівництві. Серія продуктів: ArchiTECH.PC, SoftCAD.3D, SoftCAD.2D. Розроблювач – SoftCAD International, США. http://www.softcad.com

З цього розмаїття програмних комплексів вигідно відрізняється система AutoCAD. AutoCAD – це найрозповсюдженіша в нас і за кордоном інженерна система автоматизації проектування найрізноманітніших об'єктів: від плану офісу до космічних станцій.

Система AutoCAD постійно розвивається та складається з трьох основних компонентів: графічного редактора AutoCAD, мови програмування високого рівня AutoLISP та інструментальних засобів для створення графічного інтерфейсу користувача, наприклад за допомогою DCL. Цим він суттєво відрізняється від інших програмних комплексів, що дає можливість самостійно створювати графічне середовище користувача та формувати його програмне забезпечення.

ТЕМА 3 ОСНОВНІ КОМАНДИ ПОБУДОВИ ГРАФІЧНИХ ПРИМІТИВІВ В СИСТЕМІ AUTOCAD

Система AntoCAD в даний час дуже широко використовується в усьому світі для створення графічних документів в різних галузях: машинобудуванні, архітектурі, будівництві, зв'язку та інших. Цьому сприяє виняткова гнучкість системи, численні можливості налаштування і адаптації, а також інструменти розробки додатків призначених для користувачів. Система включає спеціальне програмне забезпечення для автоматизованого 2D креслення та 3D проектування, що є в даний час найбільш поширеними у світі засобами автоматизованого проектування. Розробником програми є компанія Autodesk (США) - найбільший в світі розробник та постачальник САПР для машинобудування, цивільного і промислового будівництва та інших сфер діяльності.

AutoCAD забезпечує максимальну якість проектування і випуск документації. Ефективні інструменти тривимірного моделювання дозволяють створювати практично будь-які форми, проводити дослідження різноманітних проектних ідей. При цьому можливе легке налаштування AutoCAD під вирішення власних унікальних завдань. Система вільно конфігурується з можливістю розширення функціональність можливостей до яких відносяться власні спеціалізовані додатки та використання готових рішень. Тому в AutoCAD органічно поєднуються набір ефективних функцій проектування і гнучка адаптація до конкретних вимог користувача.

Система створює не просто графічний образ а дозволяє проаналізувати сформовані об’єкти, обслуговуючи їх на зразок менеджера бази даних. При цьому забезпечується висока швидкість виконання та простота створення креслення і його модифікацій. Крім цього, AutoCAD дозволяє створювати дещо більше, ніж креслення. Він надає можливість логічно пов’язані фрагменти креслення розмістити на виділених шарах або згрупувати їх і розглядати як єдине ціле. З допомогою простих команд можливо просто та швидко поставити розмірні лінії та тексти.

3.1 Режими креслення

Команда **LAYER –** використовується для створення, редагування нових шарів для подальшого приміщення в них примітивів.

Command: _LAYER ↵

?/Make/Set/New/ON/OFF/Color/Ltype/LWeight/Plot/Freeze/Thaw:

?/ Створити / Встановити / Новий / Включити / Виключити / Колір / Тип ліній / Заморозити / Розморозити:

При використанні команди Make створюється новий шар, який ставати поточним.

Set – встановлюється шар як поточний.

New - створюється шар, але він не поточний

Freeze/Thaw – видимий або невидимий шар.

Команда Freeze видаляє примітиви шару з робочого редактора і дозволяє користувачеві тільки розморожувати шарами.
Color/Ltype – вибір кольору і типу ліній.
Виклик з AutoLISP
(command «_LAYER» «_M» «OSN» «»)
(command «_LAYER» «_M» «OSI» «_C» 7 «_L» «DAHDOT» «»)

Команда **ERASE** – видалення примітивів

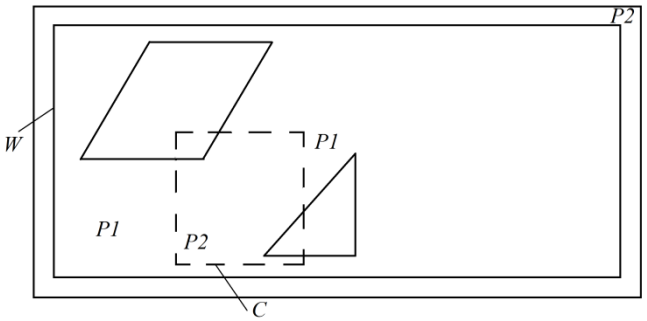
Command: ERASE ↵

Select objects < примітиви > : ↵

Ці примітиви можуть бути обрані за допомогою вибірки W, C.
Виклик з AutoLISP
(command «_ZOOM» «_E»)
(command «_ERASE» «_All» «»)

Вибір примітивів в системі AutoCAD.

Дуже багато команд AutoCAD використовують вибірку, за допомогою яких вказують примітиви, які потрібно буде редагувати.



W – вибір вікном. Вибираються всі примітиви, які повністю потрапили всередину рамки вікна.
C – до вибірки потрапляють всі примітиви, які повністю або частково захоплені рамкою.
W, C – можна регулювати вказуючи напрямок руху вікна.

Рисунок 4 – Варіанти вибору примітивів

3.2 Команди креслення

Команда **LINE –** Використовують для креслення відрізків прямих ліній.
Інвціалізація команди

Command: LINE ↵

Виклнання

From point: 1,1 ↵ P1

To point: 5,2 ↵ P2

To point: ↵

Виклик з AutoLISP
(Command «_LINE» «P1» «P2» «»)

Команда **CIRCLE –** Використовують для креслення кола.

Command: CIRCLE ↵

3P/2P/TTR/<center point>: 5,5 ↵

Diameter/<Radius>: 3 ↵

Надається можливість накреслити коло по трьом точкам, по двох точках до діаметру, по двом дотичним і радіусу, за замовчуванням по центру діаметра або радіуса.

Diameter/<Radius>: D ↵

Diameter: 6 ↵

Виклик з AutoLISP

(Command «_CIRCLE» PC 3)

(Command «_CIRCLE» PC «_D» 6)

Команда **PLINE -** Команда побудови полілінії.

Полілінія представляє собою зв'язок послідовних ламаних і дуг, які розглядаються в AutoCADe як один примітив.

Command: PLINE ↵

From point: 1,1 ↵

Arc/Close/Half width/Length/Undo/Width/<End point>:

Дуга/Замкнути/Половина ширини/Довжина/Видалити/Ширина/<Кінцева точка>/

Arc – побудова дуги по вказаним параметрам.

Close – замкнути полілінію.

Half width – половина ширини з запитом на початок і кінець лінії.

Length – вказується довжина відрізка, який спрямований туди ж, що і попередній.

Перехід в режим побудови дуги (Arc) :

Angle/CEnter/CLose/Direction/Half width/Line/Radius/Second pt/Undo/

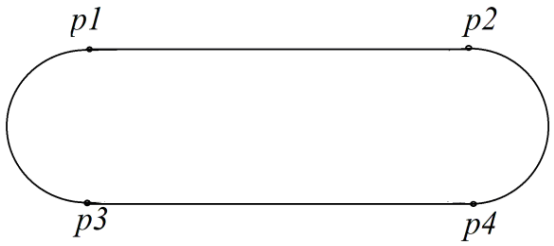
Width/<End point>:

Кут/Центр/Замкнуть/Напрямок/Половина ширини/Лінія/Радіус/По двом точкам/ Видалити / Ширина/< Кінцева точка дуги>/

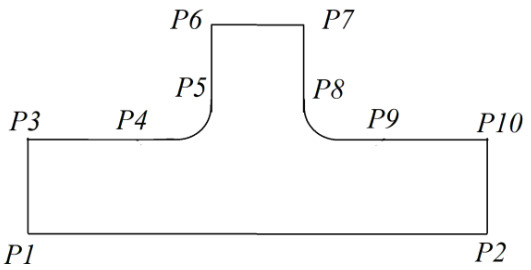
Line – повернення в попередній пункт команд. Вихід з команди: <Enter>.

В команді PLINE за замовчуванням дуга будується як дотична до попереднього примітиву. Якщо дуга не є дотичною, то її зручніше виконувати по центру дуги з урахуванням того, що за замовчуванням вона будується проти годинникової стрілки, або за напрямком.

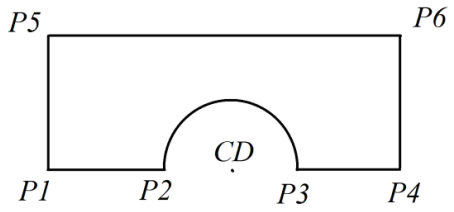
Виклик з AutoLISP



(command «_PLINE» P1 P2 «_A» P4 «_L» P5 «_A» P1 «_L» ««)



(command «_PLINE» P1 P3 P4 «_A» P5 «_L» P6 P7 P8 «_A» P9 «_L» P10 P2 «_C»)



(command «_PLINE» P1 P5 P6 P4 P3 «_A» «_CE» CD P2 «_L» P1 ««)

(command «_PLINE» P1 P2 «_A» «_D» (polar P2 (/ pi 2) 1) P3 «_L» P4 P6 P5 P1 ««)

Рисунок 5 – Побудова поліліній

Команда **PEDIT –** для редагування полілінії.

Command: PEDIT ↵
Close/Join/Width/Edit vertex/Fit/Spline/Decurve/Ltype gen/Undo/exit <x>:
Закрити/Приєднати /Ширина/Редагувати вершину/Згладити/Сплайн/
Відновити/Тип/Скасувати/Вихід/:
Виклик з AutoLISP
(Command: «_PEDIT» en «« «_S» ««)

Команда **MIRROR –** Дзеркальне відображення примітивів

Command: MIRROR ↵
Select objects <выбор объекта> : ↵
First point of mirror line <точка>: ↵
Second point <точка>: ↵
Delete old object? <N>: N ↵
Виклик з AutoLISP
(Command «_MIRROR» en «« P1 P2 «_N»)

Команда **HATCH** – виконання штрихування в замкнутій області.

Command: HATCH ↵
pattern (? or name / u, style)
шаблон(? або ім’я/u, тип)
Angle for eross hatch line:
Кут для косою штрихування:
Specing between line:
Інтервал між лініями:

Double hatch area <N>:
Подвійна штриховка
Select objects < > : ↵
Acad. pat – описані штрихування в загальному вигляді.
Виклик з AutoLISP
(command «_HATCH» «_U» 45 3 «_N» en ««)

3.3 Проставлення розмірів та тексту

Команди **DIM DIM1** використовуються для проставлення розмірів у кресленні.
Команда DIM використовується для декількох, DIM1 для 1 розміру.
Для того щоб розміри відповідали вимогам ЕСКД потрібно вказати значення опції.
DIMTXT => 5 – висота тексту
DIMASZ => 5 – розмір стрілки
DIMTAD => 1 – текст над размірною лінією
DIMTIH => 0 – безперервна розмірна лінія
DIMGAP => 1,5 – відстаней від розмірної лінії до тексту

В кресленнях зазвичай встановлюються горизонтальні, вертикальні та паралельні розмірні лінії.
HOR – горизонтальний
VER – вертикальний
AL – паралельний уведеним точкам
EXIT – вихід з команди «DIM»

1. Ввести точку першої виносної лінії
2. Ввести точку другої виносної лінії
3. Точка розмірної лінії
4. Текст

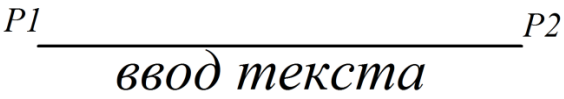
Виклик з AutoLISP
(command «_DIM»)
(command «VER» P1 P2 (polar bp pi 10) (rtos D1 2 0))
(command «HOR» P2 P3 (polar P2 (* pi 1.5) 10)(rtos L 2 0))
(command «_EXIT»)

Команда **TEXT** – призначена для введення тексту в креслення.
Command: TEXT ↵
Specify start point of text or [Justify/Style]: 10,50 ↵
Задати початкову точку або
Specify height <2.5000>: ↵
Висота
Specify rotation angle of text <0>: ↵
кут

Enter text: «ввод текста» ↵

Specify start point of text or [Justify/Style]: j ↵
Enter an option

[Align/Fit/Center/Middle/Right/TL/TC/TR/ML/MC/MR/BL/BC/BR]: F ↵
Ця опція дозволяє вказувати текст між двома точками P1 и P2.



Опція Align виконує ту ж функцію, але відрізняється тим, що символи будуть масштабуватися як по ширині так і по висоті (стиснення вирівняний по висоті щодо довжини).
текст вирівнюється:
M – по центру
B – вниз
Ввести точку 1
Ввести точку 2
Висоту тексту
Текст.

Виклик з AutoLISP
(command «_TEXT « P1 4.0 0 « введення тексту «)
(command «_TEXT « «_F» P1 P2 4.0 « введення тексту «)

Символи, які найбільше використовуються:%%c – ϕ
%%c – Ø(знак діаметра)
%%p – ± (плюс мінус)
%%d – ˚ (градус)
%%o – підкреслення
%%u – надкреслення

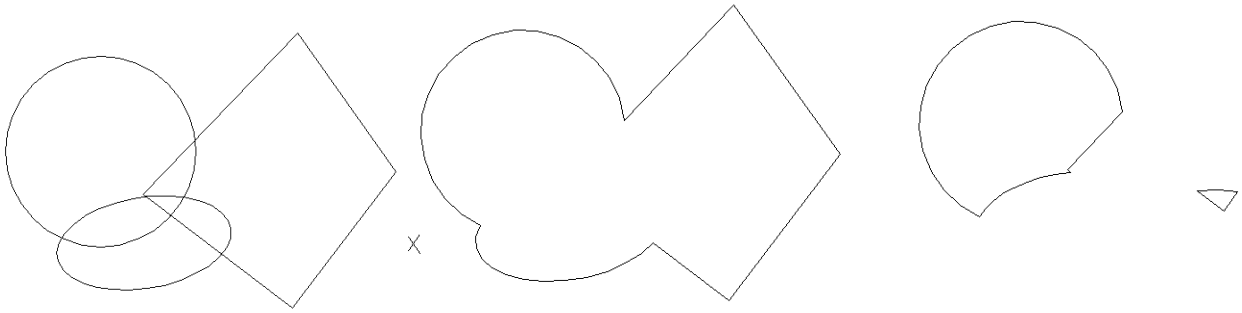
3.4 Твердотільне моделювання

Довільний плоский замкнутий контур (окружність, замкнута полілінія, відрізки у формі замкнутій ламаної й інші подібні їм об'єкти) можна зробити областю. Для цього використовується команда **REGION**, попередньо визначивши примітиви які складають область.

Область — це двовимірний об'єкт, що обмежений замкнутою границею і має внутрішність. В області можуть бути отвори. Області можуть бути об’єднаними чи складеними. Вони являються непрозорими для команд HIDE та RENDER (крім ділянок, що є отворами). Аналогом області є тонка листова деталь, у якій можуть матися вирізи. Але головне произначення області — це можливість використання її для побудови тіл складної форми.

Команда REGION вибирає об'єкти, і по закінченні їхнього вибору повідомляє про кількість створених областей. Однією командою може бути створено кілька областей, якщо зазначені користувачем об'єкти задовольняють необхідним вимогам.

Над областями можливі операції об'єднання, вирахування і перетинання. Перша з них об'єднання – **UNION**, друга вирахування – **SUBTRACT** і перетинання – **INERSECT** .



Області об'єднання вирахування перетинання

Рисунок 6 – Команди об'єднання, вирахування і перетинання області

Виклик з AutoLISP
(command «_SUBTRACT» en01 en02 «« en11 en12 en13 ««)

Тіло в системі AutoCAD можна отримати в результаті використання команд побудови стандартних форм.

Використовують наступні команди: ШУХЛЯДА – **BOX**, КУЛЯ – **SPHERE**, ЦИЛІНДР – **CYLINDER**, КОНУС – **CONE**, КЛИН – **WEDGE** і ТОР.

Також як результат обертання та видавлювання областей.
Команда **EXTRUDE –** побудова тіла методом видавлювання. При цьому потрібно:
1. Вказати об'єкт.
2. Визначити глибину введення або [траєкторія].
3. Кут видавлювання .
Виклик з AutoLISP
(command «_EXTRUDE» en «« 100 0)

Команда **REVOLVE –** побудова тіла метлдом обертання областей. При цьому потрібно:
1. Вказати об'єкт.
2. Визначити вісь або лінію відносно якої відбувається обертання.
3. Кут обертання.
Виклик з AutoLISP
(command «_REVOLVE» en1 «« «_x» 270)

ТЕМА 4 AUTOLISP – БАЗОВА МОВА ПРОГРАМУВАННЯ В AUTOCAD

Використання в системі AutoCAD мови AutoLISP не тільки значно прискорює процес розроблення проектної документації, але і дозволяє створювати нові команди графічного редактора, спеціалізовані меню в середовищі AutoCAD, здійснювати доступ до графічної бази даних і модернізувати її, розробляти функції для розв’язання найрізноманітніших задач і, крім того, створювати ефективні системи та підсистеми, пов'язані з обробкою інформації, поданої у вигляді символів і чисел. Форми подання програми й даних, отриманих нею у AutoLISP, однакові. Програми можуть обробляти, а також перетворювати інші програми і навіть самих себе, що дозволяє ефективно використовувати LISP для розв’язання широкого кола завдань.

Крім засобів виконання обчислень, AutoLISP містить функції, що надають програмам доступ до графічної бази даних поточного креслення. AutoLISP дозволяє також керувати графічним редактором AutoCAD і звертатися до його власних команд. За допомогою програм AutoLISP можна створювати доповнення, спрямовані на конкретну предметну область застосування функції, що поєднують у собі запити до користувача (діалог), можливість вибору за умовою декількох варіантів використання значень за замовчуванням. Хоча макровизначення, які створюють під час написання меню AutoCAD, можуть бути досить складними та потужними, без AutoLISP вони являють собою лише комбінації стандартних команд AutoCAD. Включенням у макровизначення меню функцій AutoLISP можна перетворити меню AutoCAD на інтелектуальний засіб автоматизації проектування.

AutoLISP дозволяє:

- використовувати змінні та вирази, відповідаючи на запити команд AutoCAD;
- читати і записувати зовнішні файли, у такий спосіб здійснюючи обмін із зовнішніми програмами, які можливо запускати з AutoCAD;
- створювати різні функції та нові команди AutoCAD, розширюючи в такий спосіб графічні можливості усієї системи;
- здійснювати програмний доступ (зчитування) до даних, які є об'єктами рисунка, а також до таблиць AutoCAD, у яких зберігається інформація про блоки, шари, види, стилі і типи ліній;
- здійснювати програмне керування графічним екраном AutoCAD і введенням/виведенням з різних пристроїв.

4.1 Функції уведення даних

(setq <ім'я1> <вираз1> [<ім'я2> <вираз2>] ...)

Функція присвоює значення «вираз1» аргументу «ім'я1», «вираз2» аргументу «ім'я2» і т. д. Вона повертає останній аргумент "вираз".

Приклад: (setq a 5.0) повертає 5.000000

(set <ім'я > <вираз >)

Функція встановлює значення аргументу «ім'я» еквівалентним аргументу «вираз» і повертає цей вираз, «ім'я» є назвою символу, перед яким іде апостроф.

Приклад: (set 'a 5.0) повертає 5.000000 і визначає символ *A*
(set (quote b) 'a) повертає *A* і визначає символ *B*

Якщо *set* використовується з ім'ям символу без апострофа, то вона може присвоїти непрямо нове значення іншому символу.

Приклад (ураховуючи зроблені вище визначення):
(set b 640) поверне 640 і присвоїть це значення змінній *A*.

(getdist [<точка>] [<підказка>])

Функція запрошує ввести відстані. «Підказка», якщо її задано, відображається в нижньому рядку екрана. «Точка», якщо її задано, є базовою точкою для відліку відстані. Відстані можна ввести з клавіатури, використовуючи поточний формат одиниці виміру. Слід враховувати, що яким би не був поточний формат, функція повертає еквівалентне дійсне число. Відстань можна ввести в графічному режимі. Для цього достатньо вказати на екрані дві точки (якщо задано «точку», то достатньо вказати тільки одну). AutoCAD відображає гумову нитку для полегшення фіксування другої точки.

Приклад: (setq dist (getdist '(1.0 3.5) "Як далеко ?"))

(getpoint [<точка>] [<підказка>])

Функція дозволяє ввести точку з клавіатури. Якщо перший аргумент задано, то на екрані з'являється гумова нитка, яка зв'язує цю точку з поточним положенням курсора.

Приклад: (setq bp (getpoint "\n Ввести bp: "))

(getreal [<підказка>])

Функція зчитує з клавіатури дійсне число і видає його. Якщо задано аргумент, то в нижньому рядку дисплея виводиться «підказка».

(getstring [<знак>] [<підказка>])

Функція приймає з клавіатури рядок. Якщо «знак» заданий і не *nil*, то всередині рядка, який вводять, можуть міститися пробіли (функція обов'язково має завершуватись <ENTER>).

Приклад: (setq s (getstring *T* "Ваше прізвище, ім'я та по батькові? "))

(princ <вираз> [<дескриптор файлу>]

Функція подібна до функції *PRINT*1*,* за винятком того, що символи керуванння у «виразі» друкують без редагування. Взагалі, *PRINT*1 призначена для того, щоб друкувати вирази зручними для користувача, а *PRINC* – для запису у файл, призначеного для зчитування такими функціями, як, наприклад, *READ-LINE*.

(print <вираз> [<дескриптор файлу>])

Ця функція аналогічна функції *PRINT1*, однак перед «виразом» виводиться новий рядок, після чого – пробіл.

4.2 Математичні функції

(+ <число> <число>...)

(– <число > <число > ...)

(*<число > <число >...)

(/<число > <число >...)

Функція повертає суму, різницю, добуток, частку всіх аргументів, які можуть бути як цілими числами, так і числами з плаваючою крапкою.

Приклад: (+ 1 2 3 4.5) результат 10.500000
(/ 100 20 2.0) результат 2.500000

(1+ <число >)

Функція збільшує «число» на 1.

Приклад: (1+ 5) результат 6

(1– <число >)

Функція зменшує «число» на 1.

Приклад: (l– –17.5) результат –18.5

(abs <число >)

Функція повертає абсолютне значення аргументу (<число>).

Приклад: (abs –99.25) результат 99.25

(atan <число1> [<число2>])

Якщо немає «числа2», то функція повертає арктангенс «числа1» в радіанах.

Приклад: (atan 0.5) результат 0.463647

Якщо задають обидва аргументи, то функція повертає арктангенс «число1»/»число2». Якщо «число2» дорівнює нулю, то повертається значення 1.57... або –1.57... залежно від знака «числа1».

Приклад: (atan 2.0 3.0) результат 0.588002

(cos <кут>)

Функція повертає косинус «кута».

Приклад: (cos 0.0) результат 1.0000000

(sin <кут>)

Функція повертає синус «кута».

Приклад: (sin 1.0) повертає 0.841471

(rem <число1> <число2> ...)

Функція ділить «число1» на «число2» і повертає остачу.

Приклад: (rem 42 12) повертає 6

(sqrt <число>)
Функція повертає квадратний корінь «числа».

Приклад: (sqrt 4) повертає 2.000000

(expr <число>)
Функція обчислює степінь числа *e*.
Приклад: (exp 1.0) результат 2.718282

(expt <основа> <показник>)
Функція підносить до степеня «показник» число «основа».
Приклад: (expt 3.0 2.0) результат 9.000000

(log <число>)
Функція повертає натуральний логарифм «числа».
Приклад: (log 4.5) результат 1.504077

4.3 Робота зі списками

(car <список>)
Функція повертає перший елемент списку. Якщо список порожній, то повертається *nil*.

Приклад: (car '(a b c)) результат a

(cdr <список>)
Функція повертає список, з якого спочатку вилучають перший елемент. Якщо список порожній, то повертається *nil*.

Приклад: (cdr '(a b c)) результат (b c)

Якщо списком є точкова пара, то *CDR* повертає другий елемент списку, який трактується як елемент, а не список.

Приклад: (cdr '(a . b)) результат *b*

caar, cadr, cddr, cadar і т. д.
В AutoLISP можлива конкатенація *CAR* та *CDR* до 4-го рівня.

Нехай має місце присвоювання:
(setq x '((a b) c d)) ,
тоді (caar x) еквівалентно (car (car x)) результат *a*
(cdar x) еквівалентно (cdr (car x)) результат (*b*)
(cadar x) еквівалентно (car (cdr (car x))) результат *b*
(cadr x) еквівалентно (car (cdr x)) результат *c*
(cons <новий перший елемент> <список>)

Функція є основним виробником списків. Вона додає «новий перший елемент» на початок існуючого списку.
Приклад: (cons 'a '(b c d)) результат (a b c d)

(lambda <аргументи> <вираз>....)
Функція дозволяє знайти безіменну функцію. Її використовують у тому разі, якщо додаткові витрати, пов'язані з отриманням цієї функції, не виправдані.

Крім того, оскільки функція визначається безпосередньо там, де її передбачається використовувати, вся програма тоді легше зчитується. *LAMBADA* повертає значення останнього зі своїх виразів і часто використовується разом з *APPLY* чи *MAPCAR* для роботи зі списками.

Приклад: (apply '(lambda (x y z) (* x (- y z))) '(5 20 14)) результат дорівнює 30

(last <список>)

Функція повертає останній елемент списку.

Приклад: (last '(a b c d e)) результат є

(list <вираз>....)

Функція формує з аргументів ("вираз") один загальний список.

Приклад: (list 'a 'b 'c) результат (a b c)

(length <список>)

Функція повертає ціле число, яке дорівнює кількості елементів у списку.

Приклад: (length '(a b c d)) результат 4

(append <вираз>...)

Функція приймає довільну кількість аргументів списочного типу і формує з них один список.

Приклад: (append '(a b) '(c d)) результат (a b c d)

(apply <функція> <список>)

Виконується виклик функції, ім'я якої є першим аргументом. Другим аргументом цієї функції є список, до якого застосовують «функцію».

Приклад: (apply '+ '(1 2 3)) результат 6

(nth <N> <список>)

Функція повертає *N*-й елемент зі «списку». Якщо *N* дорівнює нулю, то повертається перший елемент. Якщо *N* більше від кількості елементів «списку», то повертається *nil*.

Приклад: (nth 3 '(a b c d e)) результат *d*

(assoc <елемент> <a_список>)

Функція шукає у списку «a_список» підсписок, який має «елемент». Якщо такий підсписок не знайдено, то видається *nil*.

Приклад: нехай список *AL* визначено так:

((name box) (width 3) (size 4.72) (depth 5)) ,

тоді (assoc 'size al) результат (size 4.72)

(assoc 'weig al) результат *nil*

(mapcar <функція> <список1>...<список N>)

Функція повертає результат виконання «функції», аргументи якої вибирають зі списків «список1», «список2» і т. д. Кількість списків має відповідати кількості аргументів «функції».

Приклад: (mapcar '1+ '(10 20 30)) результат (11 21 31)

(mapcar '+ '(10 20 30) '(4 3 2)) результат (14 23 32)

За допомогою *LAMBDA* визначають безіменну функцію, виконати яку можна, скориставшись *MAPCAR*. Подібний засіб доцільно використовувати, коли деякі аргументи функції є константами.

Приклад: (mapcar '(lambda (x) (+x 3)) '(10 20 30)) результат (13 23 33)

(reverse <список>)

Функція повертає аргумент «список» з елементами, розташованими у зворотному порядку.

Приклад: (reverse ' ((a) b c)) повертає (C B (A))

(strcat <рядок1> <рядок2> ...)

Функція повертає рядок, який є конкатенацією аргументів «рядок1», «рядок2» і т. д.

Приклад: (strcat "a" "bout") повертає "about"

(subst <новий> <старий> <список>)

Функція шукає у «списку» аргумент «старий» (старий елемент) і повертає копію «списку» з аргументом «новий» (новий елемент) замість кожного входження старого елемента. Якщо аргумент «старий» не знайдено у «списку», функція *SUBST* повертає список непереробленим.

Приклад: (setq sample ' (a b (c d) b)) ,
тоді (subst 'qq 'b sample) повертає (A QQ (C D) QQ)

Якщо функцію *SUBST* використовують разом з *ASSOC*, то вона забезпечує доцільну заміну значення, пов'язаного з ключовим словом в асоціативному списку.

(substr <рядок> <початок> [<довжина>])

Функція повертає підрядок «рядка», починаючи з позиції «початок» у «рядку». Довжина підрядка визначається аргументом «довжина». Якщо аргумент «довжина» не заданий, то виділяється підрядок з указаної позиції і до кінця рядка.

Приклад: (substr "abcde" 2 1) повертає "b"

4.4 Логічні функції

(= <число> <число>...)

Функція виконує перевірку «на дорівнює» і повертає *T*, якщо всі аргументи чисельно дорівнюють один одному, інакше – повертає *nil*. Функція порівнює і текстові рядки.

Приклад: (= 20 388) результат *nil*
(= 2.4 2.4 2.4) результат *T*

(/= <число 1> <число 2>)

Функція є виразом відношення «не дорівнює». Її визначають тільки для двох

аргументів.

Приклад: (/= 10 20) результат *T*
(/= 5.43 5.43) результат *nil*

(< <число> <число>...)

(<= <число> <число> ...)

(>< число> <число>...)

(>= <число> <число>...)

Функції є виразами відношень «менше», «менше або дорівнює», «більше», «більше або дорівнює». Якщо умови виконуються, то функція повертає *T*.

Приклад: (>= 120 17) результат *T*
(>= 57 57) результат *T*

(and <вираз>...)

Функція виконує операцію логічного додавання аргументів («виразів»). Якщо значенням хоча б одного виразу є *nil*, то функція повертає *nil*, у протилежному випадку – повертає *T*.

Приклад: (setq a 103)
(setq b nil)
(setq c "string") ,
тоді (and 1.4 a b) повертає *T*
(and 1.4 a b c) повертає *nil*

(not <елемент>)

Функція повертає *T*, якщо значенням «елемента» є *nil,* і *nil* – у протилежному випадку. Як правило, функцію *NULL* використовують для роботи зі списками, а *NOT* – для роботи з даними інших типів.

Приклад: (setq a 123)
(setq c nil) ,
тоді (not a) результат *nil*
(not c) результат *T*

(or <вираз> ...)

Функція повертає результат логічного додавання списку виразів. Якщо всі вирази мають значення *nil*, то *OR* повертає *nil*, у протилежному випадку – повертає *T*.

Приклад: (or nil 'a '()) повертає *T*
(or nil '()) повертає *nil*

(eq <вираз1> <вираз2>)

Функція з'ясовує еквівалентні вирази 1 і 2. *EQ* повертає *T*, якщо вирази дорівнюють один одному і *nil* – у протилежному випадку.

Приклад: (setq f1 '(a b c))
(setq f2 '(a b c))

(setq f3 f2)
тоді (eq f1 f3) результат *nil* (f1 і f3 не той самий список)
(eq f3 f2) результат T (f1 і f3 один і той самий список)

(equal <вираз1> <вираз2> <точність>)
Функція визначає, чи дорівнює «вираз1» «виразу2».

Приклад: (setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2) ,
тоді (eq f1 f3) результат T (f1 і f3 є однаковими списками)
(eq f2 f2) результат *T* (f1 і f3 один і той самий список)

4.5 Функції для розгалуження програм

(cond (<перевірка1> <результат1>)...)
Функція припускає використання довільної кількості елементів списочного типу. Обчислюється перший елемент списку (у тому порядку, в якому ці списки задано), якщо цей елемент не *nil*, то обчислюється вираз, що складається з інших елементів, і повертається його значення. Якщо маємо декілька виразів, то повертається значення останнього, а якщо виразів взагалі немає, тобто список містить один елемент, то повертається значення цього елемента.

Приклад: (cond ((= s "Y") 1) ((= s "N") 0) (t nil))

Функція перевіряє склад символу *s*. Якщо значення дорівнює *Y* або *y*, то виробляється 1, якщо *n* або *N*, – то 0, в інших випадках – *nil*.

(if <умова> <вираз1> [<вираз2>])
Функція визначає, виконується умова чи ні. Якщо «умова» не дорівнює *nil*, то обчислюється «вираз1», інакше – «вираз2». «Вираз2» не є обов'язковим. *IF* повертає значення вибраного виразу.

Приклад: (if (= 1 3) "ТАК" "НІ") результат "НІ"
(if (= 2 (+ 1 3) "ТАК") результат *nil*

(progn <вираз> ...)
Функція послідовно обчислює кожний вираз і повертає значення останнього виразу. Можна використовувати функцію *PROGN* для обчислювання декількох виразів там, де припускається обчислювати тільки один.
Приклад: (if (= a b) (progn (setq a (+ a 10) b (- b 10))))
Якщо умовний вираз має значення, відмінне від *nil*, то функція *IF*, звичайно, обчислює тільки один вираз. У цьому прикладі *PROGN* використовували для обчислювання двох виразів.

4.6 Функції організації циклів

(foreach <ім'я> <список> <вираз>...)
Функція здійснює послідовне підставлення елементів списку замість імені і обчислює вказані вирази, кількість яких може бути довільною. *FOREACH*

повертає результат обчислення останнього виразу.

Приклад: (foreach n '(a b c) (print n))
еквівалентно: (print a) (print b) (print c)

(repeat <число> <вираз> ...)
Функція обчислює кожний аргумент «вираз» стільки разів, скільки вказується аргументом «число», і повертає значення останнього виразу.

Приклад: (setq b 100) ,
тоді (repeat 4
(setq b (+b 10))) повертає 140

(while <перевірка> <вираз> ...)
Функція обчислює аргумент «перевірка» і, якщо значення його не дорівнює *nil*, обчислює інші «вирази», а потім знову «перевірку». Це триває доти, доки значення «перевірки» не буде дорівнювати *nil*. Функція *WHILE* повертає останнє значення останнього аргументу «вираз».

Приклад, якщо (setq a 1) ,
то (while (<= a 10)
(some-func a)
(setq a (1+ a))) результат: 11

4.7 Виклик програм з AutoCAD

(command <аргументи>...)
Функція здійснює запуск команд AutoCAD з AutoLISP.

Приклад: (setq pt1 '(1.45 3.21))
(setq pt2 (getpoint "Enter a point"))
(command "_line" pt1 pt2 "")

Перші дві функції встановлюють дві точки pt1 і pt2, потім викликається команда *LINE* і ці точки використовуються як відповіді на її підказки. Символ "" еквівалентний введенню пробілу з клавіатури.

(vl-cmdf [<параметр1> [<параметр2> ... [<параметри>] ... J])
Аргументами можуть бути будь-які вирази, які потрібно передати в командний рядок AutoCAD (у тому числі команди, опції команд, дані, вирази LISP і т.д.).
На відміну від функції command, параметри функції vl-cmdf можуть містити вираження LISP, що містять функції інтерактивного введення (getint та інші). Значення, що функція повертає - T, якщо всі вирази, передані в командний рядок, виконалися без помилок, і nil - при виявленні помилок.

Приклад: (setq pt1 '(1.45 3.21))
(setq pt2 (getpoint "Enter a point"))
(vl-cmdf "_line" pt1 pt2 "")

4.8 Функції перетворення рядків і перевірки типів даних

(atof <рядок>)
Функція перетворює рядок на дійсне число.

Приклад: (atof "97.1") результат 97.10000

(atoi <рядок>)
Функція перетворює рядок на ціле число.

Приклад: (atoi "97") результат 97

(fix <число>)
Функція повертає результат перетворення "числа" до цілого типу.

Приклад: (fix 3.7) результат 3.000000

(float <число>)
Функція повертає результат перетворення "числа" до дійсного типу.

Приклад: (float 3.7) результат 3.700000

(itoa <ціле>)
Функція повертає результат перетворення «цілого» на рядковий формат.

Приклад: (itoa 33) результат "33"

(listp <елемент>)
Функція повертає *T*, якщо аргумент є списком і *nil* – у протилежному випадку.

Приклад: (listp'(a b c)) результат *T*
(listp '(a)) результат *nil*

(max <число> <число>...)
Функція повертає максимальне з "чисел"-аргументів.

Приклад: (max – 88 19 5 2) результат 19

(min <число> <число>...)
Функція повертає мінімальне із заданих "чисел".

Приклад: (min 638 – 10.0) результат –10.000000

(null <елемент)
Функція повертає *T*, якщо аргумент «елемент» має значення *nil*, і *nil* – у протилежному випадку.

Приклад: (setq a 123) ,
тоді (null a) повертає *nil*
(setq c nil)
(null c) повертає *T*

(rtos <число> [<режим> [<точність>]])
Функція повертає рядок, який складається з аргументу «число» відповідно до значень аргументів «режим», «точність» і системної змінної *DIMZIN* системи AutoCAD.

Допустимі значення аргументу «режим» такі (табл. 1):

Таблиця 1

Режим	Формат редагування
1	Науковий

2	Десятковий
3	Технічний (фути і десяткові частки дюймів)
4	Архітектурний (фути і дробові дюйми)
5	Вільні дробові одиниці

Приклад: (rtos 17.5 1 4) повертає "1.7500E+01"
(rtos 17.5 2 2) повертає "17.50"
(rtos 17.5 3 2) повертає "1'-5.50""
(rtos 17.5 5 2) повертає "17 1/2"

(strlen <рядок>)

Функція повертає довжину рядка (кількість символів).

Приклад: (strlen"abcd") повертає 4

4.9 Функції для роботи з геометричним описом об'єктів

(polar <точка> <кут> <відстань>)

Функція повертає точку, яка знаходиться на заданій відстані «відстань» від точки «точка», і ту, яка лежить на промені, що складає кут «кут» з базовим напрямом. Точка є списком двох дійсних чисел, а кут виражається в радіанах.

Приклад: (polar '(1.0 1.0) 0.785398 1.414214) повертає (2.0 2.0)

(angle <pt1> <pt2>)

Функція обчислює кут між двома точками «pt1» і «pt2» двовимірного простору (точку подано у вигляді двохелементного списку).

Приклад: (angle '(1.0 1.0) '(1.0 4.0)) результат 1.570796

(distance <точка1> <точка2>)

Функція повертає відстань між двома точками.

Приклад: (distance '(1.0 2.5) '(7.7 2.5)) результат 6.7

(inters <точка1> <точка1> <точка3> <точка4> [<>])

Функція визначає точку перетину двох ліній. "Точка1" і "точка2" є кінцями відрізка першої лінії, а «точка3» і «точка4» – другої лінії.

Приклад: (setq a '(1.0 1.0) b '(9.0 9.0))
(setq c '(1.0 1.0) d '(9.0 9.0)) ,
тоді (insert a b c d) результат *nil*
(insert a b c d nil) результат (4.000000 4.000000)

4.10 Завантаження, створення функцій та оброблення помилок

(load <ім'я_файлу>)

Функція завантажує файл, складений з виразів AutoLISP, та обчислює ці вирази. "Ім'я_файлу" – це ім'я файлу в системі. Розширення *LSP* друкувати не треба. Якщо файл знаходиться у другому каталозі, то замість імені можна задати весь шлях.

Приклад: "function/test1".

Якщо операція завантаження пройшла вдало, то обчислюється ім'я останньої

функції з файлу. Якщо ні, то повертається ім'я файлу. Нехай, наприклад, у файлі "/fred/test1.lsp задано *DEFUN*, яка визначає функцію *MY-FUNK*, а файл test2 не існує.

Приклад: (load"/fred/test1") результат my-func
(load "test2") результат "test2"

(defun <символ> <список аргументів> <вираз>...)

DEFUN визначає функцію з іменем «символ». Після імені функції йде список її аргументів (можливо, порожній). За цим списком можуть розміщуватися локальні параметри функції, які відокремлюють від списку похилою рискою (з обох боків від похилої риски має бути по одному пробілу). Якщо списку аргументів і параметрів немає, то дужки все одно слід ставити.

Приклад: (defun myfunc (x y)...) функція з двома аргументами
(defun myfunc (/ x y)...) функція з двома локальними символами.

За списком аргументів і локальних параметрів (символів) записуються вирази, які складають тіло функції.

DEFUN повертає значення функції, котру вона визначає.

Приклад: (defun add10 (x) (+ 10 x))
результат add10
add10 5 результат 15
add10 -7.4 результат 2.600000

Якщо ім'я функції *C:XXX*, то можна розширити AutoCAD, додавши до нього нові функції за допомогою конструкції *DEFUN*. Такі функції задовольняють вимоги:

1. Ім'я функції повинно мати формат *C:XXX*. XXX – ім'я нової команди, яке не може збігатися з уже існуючим іменем.
2. Список аргументів команди має бути порожнім, але визначення локальних символів дозволяється.

Функцію, котру визначено як команду, можна викликати просто введенням *XXX* у відповідь на підказку AutoCAD Command:

4.11 Функції файлових входів/виходів

(findfile <файл>)

Функція шукає файл по імені (короткому або повному). Якщо аргумент містить ім'я файлу без повного шляху, то пошук виконується у робочому каталозі і по стандартних для Autocad шляхам пошуку. Повертає повне ім'я файлу або nil, якщо файлу немає.

Приклад: (findfile "data.txt")

(getfiled <заголовок> <ім'я > <расширення> <прапорець>)

Функція Викликає діалогове вікно пошуку файлу.

Заголовок – назва діалогового вікна

Ім'я – ім'я файлу або теки, з якою починається пошук

Розширення – розширення файлу. («« замінюється на «*»)

Прапорець – опція функції яка має значення

1 – при створенні нового файлу (не можна використовувати для вибору існуючого файлу)

4 – дозволяє вводити ім'я файлу з будь-яким розширенням (або без нього)

16 – аргумент ім'я трактується як ім'я каталогу, в якій треба шукати файл

Приклад: (setq a (open "file.ext" "r"))

(getfiled «Виберіть файл» «E:/stud» «doc» 16) вибирається каталог для пошуку файлу

(getfiled «Збережіть файл» «E:/stud» «doc» 17) вибирається каталог для збереження файлу без вказівки імені файлу в полі діалогу

(getfiled «Збережіть файл» «E:/stud/rez» «doc» 1) вибирається каталог для збереження файлу з вказівкою імені файлу в полі діалогу

(open <ім'я файлу> <режим>)

Функція відкриває файл для забезпечення доступу за допомогою функцій введення/виведенння AutoLISP. Вона повертає дескриптор файлу, який має використовуватись іншими функціями введення/виведення; тому результат виконання даної функції присвоюється символу.

Приклад: (setq a (open "file.ext" "r"))

Допустимі режими описано в табл. 2.

Таблиця 2

Режим	Опис
"*r*"	Відкритий для зчитування. Якщо «ім'я файлу» не існує, то повертається *nil*
"*w*"	Відкритий для запису. Якщо файл не існує, то створюється і відкривається новий. Якщо існує, то його дані будуть втрачені
"*a*"	Відкритий для доповнення. Якщо файл не існує, то створюється і відкривається новий файл. Якщо існує, то він відкривається і нові дані розміщують наступними за тими, які зберігаються у ньому. Отже, реалізується доповнення даних у файл

«Ім'я файлу» може містити префікс директорії.

Приклад: (setq f (open "/x/new.tst" "w")) повертає дескриптор файлу

(close <дескриптор>)

Функція закриває файл і повертає *nil*. Нехай *X* є дескриптором відкритого файлу, тоді

(close X) повертає *nil* і закриває файл .

(read-line [<дескриптор файлу>])

Функція зчитує рядок, який уведено з клавіатури або з відкритого файлу, заданого своїм дескриптором «дескриптор файлу». Якщо трапляється кінець

файлу, то *READ-LINE* повертає *nil*, у противному разі – повертає зчитаний рядок.

(read <рядок>)

Функція повертає перший список або «атом», одержаний з рядка. У рядку не має бути пробілів.

Приклад: (read "(2 4 1)") повертає (2 4 1)

4.12 Системні та спеціальні функції

Система AutoCAD має різні попередні змінні, багато з яких можуть використовуватись для зміни різноманітних режимів і меж креслення. Доступ до системних змінних здійснюється за допомогою функцій *GETVAR* і *SETVAR*.

(getvar <ім'я-змінної>)

Функція дозволяє одержати від AutoCAD значення системної змінної. Ім'я змінної необхідно взяти у подвійні лапки. Нехай, наприклад, радіус спряження дорівнює 0,25 (значення змінної *FILLETRAD* дорівнює 0,25).

Приклад: (getvar "FILLETRAD") результат 0.250000

(setvar <ім'я-змінної> значення)

Присвоює задане значення вказаній системній змінній. Ім'я змінної слід узяти у подвійні лапки (у дод. 1 наведено список системних змінних AutoCAD).

(eval <вираз>)

Функція розраховує «вираз» і повертає його значення. Вираз є будь-якою допустимою у *LISP* конструкцією.

Приклад: (eval (abs -10)) результат 10
(setq a '(+ 3 5)) (eval a) результат 8

4.13 Доступ до примітивів і пристроїв

4.13.1 Спеціальні типи даних

(ssget [<режим>] [<точка1> [точка2>][<список точок>] [<фільтр-список>])

Функцію використовується для організації набору та вибору. Аргументи «точка1» і «точка2» є списками, котрі задають точки вибору. Задання точки без аргументу «режим» рівносильне вибору примітиву вказуванням точки. Якщо всі аргументи пропускаються, то *SSGET* через AutoCAD видає користувачу загальний запит *Select objects*:, дозволяючи інтерактивну конструкцію наборів вибору. Додатковий аргумент «режим» є рядком, котрий задає тип набору примітиву, що виконується. Цим типом може бути "*_W*", "*_C*", "*_F*","*_L*","*_P*" або "*_X*", чи "*_I*".

Приклади:

- (ssget) видає стандартний запит Select objects: (Виберіть об'єкти :) і потім

створює набір у відповідності з інтерактивними діями користувача

- (ssget '(125.4 58.1)) створює набір з усіх примітивів, що проходять через точку (125.4 58.1)
- (ssget '(125.4 58.1)' ((0. "POLYLINE"))) створює набір з усіх примітивів типу POLYLINE, що проходять через точку (125.4 58.1)
- (ssget "_C" '(14.3 -5.2)' (27.6 106.1)) створює набір з примітивів, обираних січною рамкою з кутами в точках (14.3 -5.2) і (27.6 106.1)
- (ssget "_F" '((1.0 0.0) (2.0 1.0) (10.0 1.0) (12.0 0.0)(1.0 0.0))) створює набір з примітивів, що перетинаються ламаною лінією, що має чотири вершини
- (ssget "_X" (list (cons 0 "LWPOLYLINE") (cons 8 "SET"))) створює набір з усіх примітивів типу LWPOLYLINE, розміщених на шарі SET
- (ssget "_X" '((0. "LWPOLYLINE") (8. "SET, ACR, BELL"))) створює набір з усіх примітивів типу LWPOLYLINE, розміщених на шарах SET, ACR і BELL
- (ssget "_X" '((0. "*LINE"))) створює набір з усіх примітивів малюнка, що є об'єктами типу * LINE (тобто LINE, SPLINE, POLYLINE, LWPOLYLINE)
- (ssget "_X" '((8. "СТЕНА [l-3]"))) створює набір з усіх примітивів рисунка на шарах СТЕНА1, СТЕНА2, СТЕНА3

(sslength <нв>)

Функція повертає ціле число, що містить номери примітивів у наборі вибору «нв».

(ssname <нв> <індекс>)

Функція повертає ім'я примітиву елемента «індекс» набору вибору «нв». Якщо аргумент «індекс» від'ємний або більше від номера останнього примітиву в наборі вибору, то повертається *nil*. Перший елемент у наборі має нульовий номер. Імена примітивів у наборах вибору, одержані за допомогою функції *SSGET*, завжди будуть іменами основних примітивів. Підпримітиви (атрибути блоків і верхівки поліліній) повертатися не будуть (доступ до них забезпечує *ENTNEXT*).

(ssadd [<ім'я> [<нв>]])

Якщо функцію можна викликати без аргументів, *SSADD* створює порожній набір вибору; якщо з аргументом імені одного примітиву створює новий набір вибору, що містить це ім'я примітиву. Якщо функцію викликають з іменем примітиву та набором вибору, то вона додає іменований примітив до набору вибору. *SSADD* завжди повертає новий або модифікований набір.

(ssdel <ім'я> <нв>)

SSDEL стирає ім'я примітиву з набору вибору «нв» і повертає ім'я набору вибору «нв». Зверніть увагу на те, що примітив фізично стирається з набору вибору та повертається новий набір без указаного елемента. Якщо в наборі вибору немає примітиву, то повертається *nil*.

(ssmemb <ім'я> <нв>)

Функція перевіряє, чи є ім'я примітиву «ім'я» членом набору вибору «нв». Якщо так, то *SSMEMB* повертає ім'я примітиву «ім'я». В іншому разі *nil*.

ЗАУВАЖЕННЯ

З допомогою функції ssget відбувається вибір лише об'єктів видимих на аркуші поточного креслення. Тільки (ssget "X") вибере всі об'єкти, включаючи ті що знаходяться за межами видового екрану.

У кресленні одночасно може бути відкрито не більше 128 наборів. При досягненні такої межі функція ssget відмовляється створювати наступні набір і повертає для них nil. Щоб видалити непотрібні набори, слід присвоїти значення nil змінним цих наборів.

4.13.2 Функції імені примітиву

(entnext [<ім'я>])

Якщо ця функція викликається без аргументів, то вона повертає ім'я першого невидаленого примітиву у базі даних. Якщо функцію *ENTNEXT* можна викликати з аргументом імені примітиву «ім'я», то вона повертає ім'я першого невидаленого примітиву, котрий є наступним за примітивом «ім'я» у файлі креслення. Якщо у файлі немає такого примітиву, то повертається *nil*.

(entlast)

Функція повертає ім'я останнього невидаленого головного примітиву у файлі креслення. Її часто використовують для отримання імені нового примітиву, котрий додано за допомогою функції *COMMAND*. Примітив не обов'язково має бути на екрані чи на «розмерзлому» шарі.

(entsel [<підказка>])

Іноді, працюючи з примітивами, бажано одночасно вибирати примітив і задавати точку, за якою він вибирався. Прикладом цього може бути використання режимів об'єктної фіксації та команд *BREAK, TRIM* і *EXTEND* системи AutoCAD. Функція *ENTSEL* дає можливість програмам AutoLISP виконувати цю операцію.

4.13.3 Функції обробки атрибутів примітивів

Функції, наведені у цьому розділі, дозволяють шукати і модифікувати дані, що є примітивами AutoCAD. У всіх функціях використовуються імена примітивів для ідентифікації конкретних даних.

(entdel <ім'я>)

Примітив з іменем «ім'я» вимикають з креслення, якщо він там є. Якщо примітив вимкнули з креслення раніше у даному сеансі, то функція повертає його. Вимкнення примітивів з файлу креслення відбувається під час виходу з графічного редактора, тому їх повернення за допомогою *ENTDEL* можливо тільки у рамках того сеансу, у якому вони були вимкнені.

(entget <ім'я>)

Функція шукає примітив «ім'я» у файлі креслення та повертає список, котрий є описом цього примітиву. Зверніть увагу на те, що в підсписок, який містить

координати точки, не входить символ «точка». Оскільки точка є двохелементним списком, то вся група має бути трьохелементним списком. Функція *CDR* має завжди повертати трьохелементний список, а символ «точка» доповнює список до необхідної довжини.

(entmake [elist])

Функція може визначати як графічні, так і неграфічні об'єкти. «elist» – список об'єктів визначення даних, який задається у форматі, подібному до формату функції *ENTGET*. Аргумент *ELIST* має містити всю інформацію, необхідну для визначення об'єкта. Якщо будь-які необхідні визначення даних пропущені, функція *ENTMAKE* повертає *nil* і не виконується. Якщо пропущено необов'язкові дані (наприклад, шар – "*layer*"), *ENTMAKE* використовує значення за замовчуванням.

Приклад: наступний код створює червоне коло (колір 62), з центром у (4,4) і радіусом.

Необов'язковий шар і тип лінії поля були опущені і тому набувають значень за замовчуванням.

Command: (entmake '((0 . "CIRCLE") (62 . 1) (10 4.0 4.0 0.0) (40 . 1.0))
((0 . "CIRCLE") (62 . 1) (10 4.0 4.0 0.0) (40.1))

(entmod <список>)

Функції *ENTMOD* передається список у формі, котру генерує функція *ENTGET*. *ENTMOD* робить модифікацію цього списку та відповідне відновлення файлу креслення. Механізм оновлення файлу креслення за допомогою AutoLISP може бути наступною. Спочатку за допомогою *ENTGET* проводиться пошук і вилучення списку примітиву, потім цей список модифікується за допомогою AutoLISP, функцією *ENTMOD* список запам'ятовується у файлі креслення.

Існує ряд обмежень на можливості модифікації примітивів. По-перше, не змінюйте тип примітиву. Усі елементи, на котрі посилаються фрагменти оброблюваного примітиву, мають бути визначені у програмі AutoCAD на момент виклику функції.

(entupd <ім'я>)

Під час модифікації вершин поліліній або атрибута за допомогою *ENMOD* на екрані не відбувається відображення виконуваних дій. Функція *ENTUPD* забезпечує повну регенерацію вказаного об'єкта.

4.13.4 Використання імен примітивів і наборів виборів

Ім’я примітиву або набір виборів є допустимою вхідною інформацією, що передається з AutoLISP у AutoCAD, у відповідь на запрошення вказати примітиви. Як результат, примітиви, вибрані у AutoLISP, можуть оброблятися AutoCAD. На запрошення *Select objects* AutoLISP може відповісти передачею імені конкретного примітиву або передачею набору виборів. Перший випадок рівносильний указанню цього примітиву з екрана, а у другому випадку в AutoCAD передаються усі примітиви набору. Передача в AutoCAD імені

примітиву або набору виборів допустима тільки у тих випадках, коли можна було б використати опцію *LAST* процесу казання на елементи. У всіх випадках примітиви не обов'язково мають бути видимими.

4.13.5 Доступ до елементів таблиць

Таблицями символів AutoCAD є таблиці шарів, типів ліній, видів, описів блоків і текстових стилів. Функції *TBLNEXT* та *TBLSEARCH* забезпечують читання інформації з цих таблиць.

(tblnext <ім'я таблиці> [<початок>]

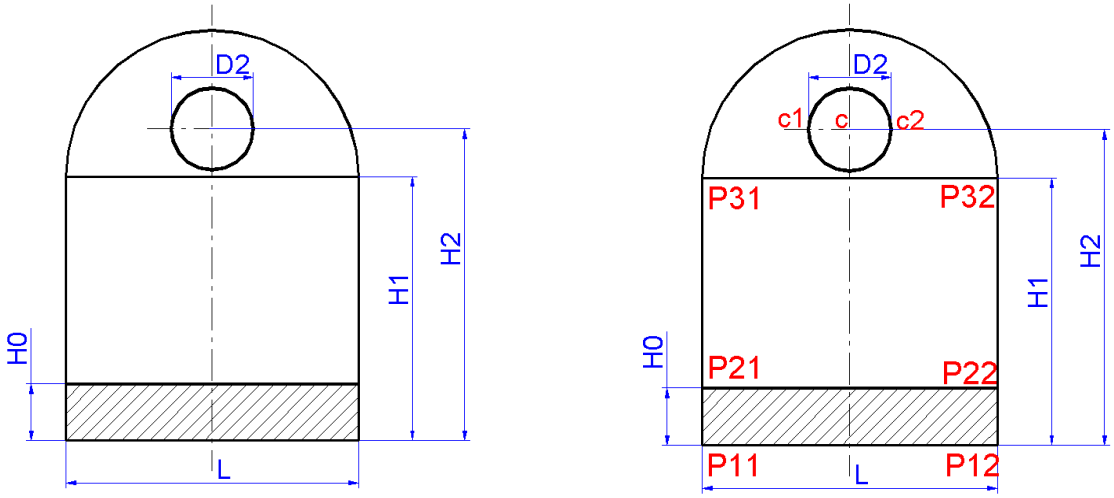
Функцію використовують для повного перегляду таблиці символів. Допустимими іменами таблиці є:

LAYER – шарів; *LTYPE* – типів ліній; *VIEW* –видів; *STYLE* –текстових стилів; *BLOCK* – опису блоків; *UCS* –систем координат користувача; *DIMSTYLE* – стилів розмірів; *APPID* – імен додатків.

(tblsearch <ім'я таблиці> <символ>)

Функція шукає у таблиці символів символ, що має задане ім'я. Імена таблиці та символу перетворюються до верхнього регістра автоматично. Якщо пошук закінчується успішно, то символ видається у такому ж форматі, як у функції *TBLNEXT*. В іншому разі – видається *nil*. Наприклад: (tblsearch "style" "standart") шукає у таблиці текстових стилів. Функція *TBLSEARCH* не впливає на стан виведення елементів за допомогою *TBLNEXT*.

4.14 Приклад виконання завдання



Завдання Розрахункова схема

Рисунок 7 – Приклад виконання програми на **AutoLISP**

Програма на функціональній мові AutoLISP

```
(defun c:Bl_25()
; Введення даних
  (setq bp (getpoint "\n Ввести Bp: "))
```

```
(setq L 50.0 H0 17.0 h1 70.0 h2 80.0 d2 10.0)
;  Визначення координат
 (setq p11 bp
       p12 (polar p11 0 l)
       p21 (polar p11 (/ pi 2) h0)
       p22 (polar p21 0 l)
       p31 (polar p11 (/ pi 2) h1)
       p32 (polar p31 0 l)
       c (list (+(car bp) (/ l 2.0)) (+(cadr bp) h2))
       c1 (polar c 0 (* d2 0.5))
       c2 (polar c pi (* d2 0.5))
       p112 (polar p11 0 l/2)
       p312 (polar p112 (/ pi 2) (+ h1 (* l 0.5)))
 )
; Креслення контура
(command "_LAYER" "_M" "OSN" "")
(command "_pline" p11 p12 p22 p21 "_c")(setq en1 (entlast))
(command "_pline" p21 p31 "_a" p32 "_L" p22 "")
(command "_pline" p31 p32 "")
(command "_pline" c1 "_a" "_ce" c c2 c1 "")
; Штрихування
 (command "_LAYER" "_M" "SHCH" "_C" 1 "" "")
(command "_HATCH" "_U" 45 5 "_n" en1 "")
; Вісь
(command "_LAYER" "_M" "OSI" "_C" 5 "" "_L" "dashdotx2" "" "")
(setvar "LTSCALE" 10)
(command "_LINE" (polar p112 (* pi 1.5) (* (distance p112 p312) 0.02))
                 (polar p112 (* pi 0.5) (* (distance p112 p312) 1.02)) "")
  ; Розміри
(command "_LAYER" "_M" "TEXT" "_C" 3 "" "")
(setvar "dimtxt" 5)(setvar "dimasz" 5)(setvar "dimtad" 1)
(setvar "dimtih" 0)(setvar "dimgap" 1.5)
(command "_dim")
(command "hor" p11 p12 (polar p11 (* pi 1.5) 10) (rtos L 2 0))
(command "hor" c1 c2 (polar c (* pi 0.5) (* d2 0.6)) (rtos d2 2 0))
(command "ver" p11 p21 (polar p11 pi 8) (rtos h0 2 0))
(command "ver" p12 p32 (polar p12 0 10) (rtos h1 2 0))
(command "ver" p12 c (polar p12 0 20) (rtos h2 2 0))
(command "_exit")
(princ))
```

ТЕМА 5 DCL – МОВА КЕРУВАННЯ ДІАЛОГОМ

Діалогові вікна в AutoCAD визначаються текстовими файлами, написаними мовою DCL. Ці файли мають розширення «.dcl» і містять опис способу відображення вікна на графічному моніторі та його склад: клавіші, списки, ковзні шкали, кнопки вибору тощо. Правила конструювання діалогових вікон задають обмеження на розмір і місце розташування. Розташування елементів вікна дуже схоже на розташування абзаців у сформатованому тексті, тому не потрібно задавати точні координати фрагментів вікон.

Діалогове вікно з меню AutoCAD можна викликати через функцію AutoLISP чи зовнішню функцію, які керують діалоговим вікном.

У кожному діалоговому вікні міститься одне чи кілька полів, що визначають його функції. До основних типів полів належать базові поля: клавіші, кнопки, текстові поля, ковзні шкали, поля списків, поля зображень. Поля можуть обрамляти рамки. Можна комбінувати поля, створюючи ряди і стовпчики.

Кожне діалогове вікно розглядають як деревоподібну структуру, вершина якої представляється мовою DCL як *dialog*. Керування вікнами здійснюють атрибути полів. Можна визначити нові поля (прототипи) і групи полів, що не зв’язані зі звичайними діалоговими вікнами. На прототипи можна посилатися та змінювати у разі необхідності їхні атрибути і попередньо визначені поля. Об’єднання використовують тільки для зовнішніх посилань. Їхні атрибути змінювати не можна.

Опис діалогового вікна здійснюють мовою DCL, що відбиває його деревоподібну структуру.

5.1 Створення DCL файлів

5.1.1. Синтаксис мови DCL

Мову DCL використовують для визначення нових і об’єднання існуючих полів у діалогові вікна. Нові вікна створюються визначеннями полів. Якщо визначення з’являється поза діалоговим вікном, воно є прототипом (визначенням поля) чи об’єднанням (елементами, що належать до групи), які можна використовувати в діалогових вікнах для посилань. Кожне посилання на визначення успадковує атрибути вихідного поля. Якщо посилаються на прототип, можна змінювати значення атрибутів чи додавати нові атрибути; у разі посилань на об'єднання атрибути не можуть бути змінені чи додані.

Якщо використовують кілька екземплярів полів із декількома загальними атрибутами, простіше визначити прототип, що містить загальні атрибути. Тоді в кожному посиланні на прототип можна змінювати чи додавати нові атрибути. У такому разі не потрібно перелічувати список усіх загальних атрибутів при кожному посиланні на поле. Оскільки атрибути успадковуються саме таким чином, то під час створення діалогового вікна простіше створити посилання на поля (особливо посилання на попередньо визначені поля), ніж створювати нові поля.

Коментарі. Коментарям у *DCL* файлах передують дві зворотні похилі риски – «//». Усе, що з'являється між «//» і кінцем рядка, ігнорується. *DCL* також дозволяє використовувати коментарі, прийняті в мові C++. Вони мають форму «/*», текст коментаря – «*/». Початкові «/*» і кінцеві «*/» символи можуть знаходитися на різних рядках.

Атрибути полів визначають їх розміщення та функціональність і схожі на змінні мови програмування. Атрибут складається з імені та значення. Значення атрибутів можуть бути:

- цілими – це числова величина (ціла чи дійсна), що являє собою розмір в одиницях чи ширини висоти символів;
- дійсними – це числова величина, у якій наявність незначного нуля в цілій частині є обов'язковою;
- рядковими – це рядок тексту, взятий у лапки (« «). Якщо рядок містить лапки, перед ним слід поставити похилу риску: \». Рядок може містити інші *ESC*-послідовності:
  \» – лапки;
  \\ – зворотну похилу риску;
  \n – новий рядок;
  \t – горизонтальну табуляцію;
- зарезервованими словами – це ідентифікатор, що складається з текстових символів і починається з літери (наприклад, *true* чи *false* – зарезервовані слова, необхідні в багатьох атрибутах). Зарезервоване слово залежить від регістра: *True* не дорівнює *true*. Імена атрибутів також залежать від регістра. Додатки завжди одержують атрибут у вигляді рядка, тому, якщо як додаток використовують числові величини, вони мають бути перетворені з рядкового вигляду.

Кнопки виходу з діалогового вікна. Файл *base.dcl* забезпечує кілька об'єднань стандартних кнопок для виходу з діалогового вікна чи його видалення. Ці стандартні об'єднання варто використовувати для збереження одноманітності вигляду діалогових вікон серед різних додатків. Крім того, у версії файлу *base.dcl* ці кнопки визначені для кожної платформи з урахуванням її особливостей, так, що використання цих визначень гарантує сумісність із кожною платформою.

5.1.2 Попередньо визначені активні поля

Попередньо визначені поля безпосередньо підтримують засоби програмувальних діалогових вікон AutoCAD. Їхні визначення містяться у файлі *base.dcl* у вигляді коментарів. Під час вибору активного поля діалогове вікно сповіщає про це додаток (програму, написану мовами *Lisp* чи C++), що керує діалоговим вікном. Подібна операція називається *викликом з поверненням*. Будь-яке попередньо визначене активне поле має відповідний видимий зовнішній (наприклад, розкриття списку чи закриття діалогу при вказівці кнопки *OK*) чи внутрішній ефект. У цьому випадку виробляється код причини, зміст якого залежить від типу ініційованого поля.

BUTTON

Клавіша – Button: Поле діалогового вікна, що нагадує звичайну клавішу клавіатури. Кнопки призначені для виконання дій, що мають зовнішній ефект для користувача: закриття діалогового вікна, поява ще одного вікна і т. д. Усі діалогові вікна містять кнопку *OK* чи її еквівалент, що дозволяє виконати дії вікна, а багато вікон – кнопку скасування, що дозволяє користувачу залишити діалогове вікно без внесення яких-небудь змін.

Атрибути

label: Мітка – рядок, узятий у лапки (значення за замовчуванням немає). Визначає напис, що з'являється на кнопці.

is_default: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо *true*, кнопка є кнопкою за замовчуванням і виконується, коли користувач натискає клавішу <ENTER>. Тільки одна кнопка в діалоговому вікні може мати значення атрибута *is_default*, що дорівнює *true*.

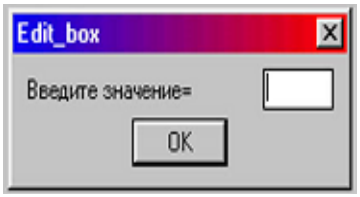
is_cancel: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення *true*, кнопку вибирають під час натискання клавіші скасування (наприклад, <ESC> чи <Ctrl+C>).

Тільки одна кнопка в діалоговому вікні може мати значення атрибута *is_cancel*, що дорівнює *true*. Кнопка, що має встановлений у *true* атрибут *is_cancel*, закриває діалогове вікно після виконання чи дії виклику з поверненням.

Приклад:

:button{ label = "Теор.чертеж"; key = "TEORCS";is_default = true;}

EDIT_BOX



Текстове поле – Edit_box: Поле, в яке користувач може вводити чи редагувати текстовий рядок. Якщо текст, який уводять, перевищує розміри текстового поля, його можна «прокрутити» у горизонтальному напрямку.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – порожні « «). Текст виводиться ліворуч від поля. Якщо він визначений, атрибут вирівнюється вліво по ширині текстового поля.

Edit_width: Можливі значення: ціле чи дійсне число. Ширина частини поля, що редагується, обмеженого рамкою текстового поля в одиницях ширини символу. Якщо атрибут не зазначено чи встановлено у нуль і ширину поля не обмежено, рамка розширюється настільки, наскільки це можливо. Якщо значення атрибута не дорівнює нулю, рамка вирівнюється вправо всередині зайнятого полем простору.

Edit_limit: Значення – ціле число (за замовчуванням – 132); максимум – 256. Максимум – максимальна кількість символів, яку може ввести користувач у

текстове поле. Коли цієї кількості досягли, драйвер монітора відкидає додаткові символи (крім <Backspase> чи <Del>) і видає сигнал.

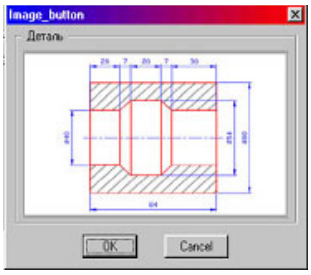
Value: Значення – рядок, узятий у лапки (за замовчуванням – порожні « «). Вихідне значення – текстове, розміщене в рамці. Значення вирівнюють уліво в редагованій частині поля.

Allow_accept: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення дорівнює *true* і користувач натиснув клавішу виконання (як правило, клавішу <ENTER>), кнопка за замовчуванням (якщо є) стає «натиснутою». Кнопкою за замовчуванням уважають ту, атрибут *is default* якої встановлений у *true*.

Приклад:

:edit_box{label="Толщ.линь";edit_width=6;key="hlin";}

IMAGE_BUTTON



Кнопка зображення – Image_button: Поле з графічним зображенням. Коли користувач вибирає клавішу зображення, програма одержує координати точки, у якій відбувся вибір. Це доцільно, якщо рисунок невеликий і вибір різних його областей має різне значення.

Атрибути

Color: Колір фону малюнка, визначений як номер кольору AutoCAD чи одне із символьних імен (за замовчуванням – 7):

dialog_line – поточний колір діалогового вікна;

dialog_forground – поточний колір символів діалогового вікна (для тексту);

dialog_background – поточний колір фону діалогового вікна;

graphic_background – поточний колір фону графічного екрана; AutoCAD (звичайно аналог 0);

black – колір AutoCAD (чорний) на чорному фоні зображується як білий;

Allow_accept: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення дорівнює *true* і користувач натиснув клавішу виконання (зазвичай, клавіша <ENTER>), клавіша за замовчуванням (якщо є) стає «натиснутою». Кнопкою за замовчуванням вважають ту, атрибут *is_default* якої встановлений у *true*.

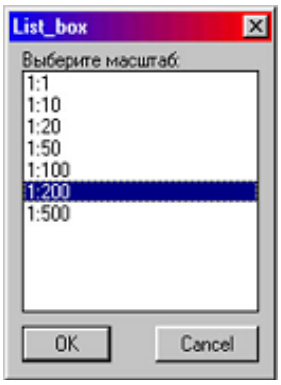
Aspect_ratio: Відношення ширини зображення до його висоти (ширина, ділена на висоту). Якщо значення дорівнює нулю (0.0), то поле зміщується під розміри зображення. Можливе значення з плаваючою точкою (значення за замовчуванням немає).

Полю зображення необхідно присвоїти точні значення атрибутів *width* і *height* чи один із цих атрибутів і значення *aspect_ratio*.

Приклад:

:image_button {key="comp1";color=0; width = 30; aspect_ratio = 0.7;}

LIST_BOX



Поле списку – List_box: Поле, що містить ряди текстових рядків. Дає можливість вибрати необхідний пункт зі списку. Як правило, список має змінну довжину, але поля списків можуть використовуватися і для списків фіксованої довжини (при виборі рядка списку вона підсвічується). Поле списку може містити рядків більше, ніж уміщує поле вікна; у цьому разі праворуч від списку поля з'являється ковзна шкала (її ввімкнено тільки тоді, коли список має більше пунктів, ніж можна ввести за один раз). Переміщуючи візир ковзної шкали можна переглянути весь список. Залежно від режиму користувач може вибрати кілька рядків списку.

Атрибути

Label: Текст, виведений над полем списку. Значення – рядок, узятий у лапки (значення за замовчуванням немає).

Multiple_select: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо *true*, у полі списку можна вибрати і підсвітити кілька пунктів. Якщо *false*, у полі списку може бути обраний тільки один пункт, а вибір іншого пункту скасує вибір попереднього.

List: Визначає початковий вибір рядків, розміщених у полі списку. Рядки відокремлюють символом нового рядка (\n). Символ табуляції (\t) може бути у кожному рядку. Значення – рядок, узятий у лапки (значення за замовчуванням немає).

Tabs: Значення – рядок, узятий у лапки, що містить цілі числа чи числа з плаваючою точкою, відокремлені пробілами (значення за замовчуванням немає). Кожне число – це величина, що визначає розташування табуляцій в одиницях ширини символу. Ці значення використовують для вертикального вирівнювання стовпців тексту в полі списку.

Value: Значення – рядок, узятий у лапки, може містити нуль («0») чи цілі числа, відокремлені пробілами (значення за замовчуванням немає). Кожне ціле (починаючи з нуля) представляє спочатку обраний пункт списку. Якщо значення атрибута *multiple_select* дорівнює *false*, атрибут *value* не може містити більше одного числа.

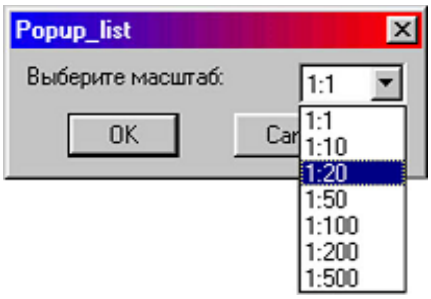
Якщо рядок порожній (« «), спочатку не буде обраний жоден пункт. У цьому разі цей атрибут можна взагалі не визначати.

Allow_accept: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо значення дорівнює *true* і користувач натиснув клавішу виконання (як правило, клавішу <ENTER>), кнопка за замовчуванням стає «натиснутою». (Кнопкою за замовчуванням уважають ту, атрибут *is_default* якої встановлений у *true*).

Приклад:

:list_box{width=20;height=35;fixed_height=true;key="list";allow_accept=true;}

POPUP_LIST



Список, що розкривається – Popup_list: Список, що розкривається, еквівалентний полю списку. Коли діалогове вікно з'являється перший раз, список, що розкривається, знаходиться в закритому стані і на екрані має вигляд кнопки з напрямленою вниз стрілкою праворуч. При виборі напису чи стрілки список розкривається й дозволяє вибрати необхідний пункт у межах вікна. Розкритий список, як правило, відображається цілком, у противному разі він буде мати ковзну шкалу праворуч, що функціонально аналогічна ковзній шкалі поля списку.

Атрибути

Label: Значення – рядок, узятий у лапки (значення за замовчуванням немає). Це текст, виведений ліворуч від списку, що розкривається. Якщо значення атрибута *label* визначено, воно вирівнюється вліво в межах поля *Popup_list*.

Edit_width: Можливі значення: ціле чи дійсне число. Ширина частини списку в одиницях ширини символу, ширина рамки, яка описує єдиний пункт, коли список, що розкривається, закритий. Якщо атрибут не зазначено чи встановлено у нуль і ширину поля не задано, рамка розширюється настільки, наскільки це можливо.

Value: Значення – рядок, узятий у лапки, що містить ціле число (за замовчуванням – 0). Ціле число (починаючи з нуля) є поточним обраним пунктом у списку (пункт, що виводиться на екран при закритому списку).

List: Значення – рядок, узятий у лапки (значення за замовчуванням немає). Визначає початковий вибір рядків, розміщених у полі списку. Рядки відкремлено символом нового рядка (\n). Символ табуляції (\t) може бути у кожному рядку.

Tabs: Значення – рядок, узятий у лапки, що містить цілі числа чи числа з плаваючою точкою відокремлені пробілами (значення за замовчуванням немає). Кожне число – це величина, яка визначає розташування табуляцій в одиницях ширини символу.

Приклад:

```
:popup_list { edit_width=15; key="OPROJ";}
```

RADIO_BUTTON

Кнопка вибору – Radio_button: Одна чи група кнопок, об'єднаних у стовпці чи ряд вибору. Кнопки вибору функціонально еквівалентні кнопкам радіопанелей: можна вибрати одну кнопку і, поки вона буде натиснута, будь-яка інша кнопка буде вимкнена. Праворуч від кнопки вибору може з'явитися необов'язкова мітка *label*.

Атрибути

Label: Текст, виведений ліворуч від кнопки вибору. Значення – рядок, узятий у лапки (значення за замовчуванням немає).

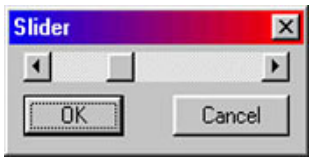
Value: Рядок, узятий у лапки (значення за замовчуванням немає). Якщо значення атрибута «1», кнопку вибору ввімкнено; якщо «0» – вимкнено; всі інші значення аналогічні «0».

Якщо з якихось причин кілька кнопок мають атрибут *value*, що дорівнює «1», буде ввімкнена тільки остання з них. Така ситуація може виникнути лише в користувацьких *DCL*-файлах.

Приклад:

:radio_button{ key="PP"; label = "сечение №1"; width = 1;}

SLIDER



Ковзна шкала – Slider: Використовують для одержання числового значення. Користувач може переміщувати візир ковзної шкали вліво чи вправо (нагору чи вниз) для одержання величини, призначення якої залежить від додатка. Ця величина повертається як рядок, що містить ціле число визначеної точності зі знаком. У додатку це значення можна масштабувати.

Атрибути

Min_value і **Max_value:** Значення – цілі числа, що визначають діапазон повороту значень ковзної шкали. Мінімальне значення за замовчуванням *Min_value* – 0. Максимальне значення за замовчуванням *Max_value* – 10 000. Діапазон має бути визначений знаковим 16-бітовим цілим, тобто від мінус 32 768 до 32 767.

Значення атрибута *Min_value* може бути більше, ніж значення *Max_value.* На деяких платформах це змінює як послідовність появи на екрані цих значень, так і послідовність їхнього повернення ковзною шкалою під час переміщення візира.

Small_increment і **Big_increment:** Значення – цілі числа, що визначають значення, які використовуються при керуванні збільшенням значення ковзної шкали. Значення за замовчуванням для атрибута *Big_increment* – десята від повного діапазону, значення за замовчуванням для *Small_increment* – сота від повного діапазону. Значення мають бути в діапазоні, обумовленому атрибутами *Min_value* і *Max_value*. Ці атрибути є необов'язковими.

Layout: Ковзна шкала може бути орієнтована горизонтально чи вертикально (за замовчуванням – горизонтально). Для горизонтальних шкал значення збільшується зліва направо, для вертикальних – знизу нагору.

Value: Значення – рядок, узятий у лапки, і який утримує поточне (ціле) значення ковзної шкали (за замовчуванням – *Min_value*).

Приклад:

:slider {key = "ms"; min_value = 8; max_value = 60; layout = vertical;
action = "0";height = 18; fixed_height = true; }

TOGGLE

Перемикач – Toggle: Оперує двійковими величинами («0» чи «1»). Його зображено невеликим прямокутником із необов'язковою міткою (*label*) праворуч. Мітка *X* з'являється чи зникає в прямокутнику за вказівкою перемикача. Перемикач дозволяє користувачу бачити й змінювати його стан «увімкнено_вимкнено».

Атрибути

Label: Текст, виведений ліворуч від кнопки перемикача. Значення – рядок, узятий у лапки.

Value: Визначає початковий стан перемикача. Значення – рядок, узятий у лапки, (за замовчуванням – «0»). Якщо значення дорівнює «0», кнопка перемикача порожня (не позначена). Якщо значення дорівнює «1», кнопка виводиться з міткою *X*.

Приклад:

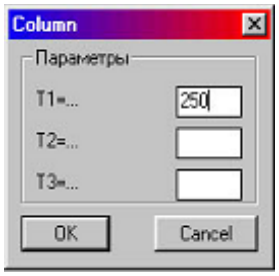
:toggle{ key = "VBB";}

5.1.3 Попередньо визначені активні групи полів

Поля можна групувати в ряди чи стовпці. Згруповані ряди й стовпці під час формування загального діалогового вікна розглядають як єдине поле, вони можуть розміщуватись у рамці та мати необов'язкову мітку (група без рамки не може мати мітку). Завдяки такому групуванню зручно розкладати поля у діалоговому вікні.

Ряд чи стовпець можна визначити для зовнішнього використання як елемент діалогового вікна. Групи називають *об'єднаннями*, оскільки вони можуть містити інші поля. Не можна змінювати атрибути об'єднання, якщо воно використовується як посилання в діалоговому вікні. У файлі *base.dcl* визначено кілька стандартних об'єднань.

COLUMN



Стовпець – Column: Стовпець чи будь-який вид поля (крім кнопки вибору), включаючи ряди й інші стовпці. Усі елементи стовпця розташовуються вертикально в порядку розташування в *DCL*-файлі.

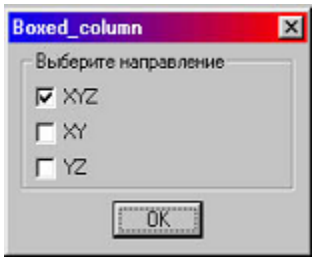
Атрибути

Стовпці без рамки не має додаткових атрибутів, крім стандартних атрибутів компонування.

Приклад:

:column{
…
}

BOXED_COLUMN



Стовпець у рамці – Boxed_column: Стовпець, що має намальовану навколо нього рамку. Діалогове вікно розташоване, як і стовпець, у рамці. Якщо стовпець має у рамці мітку, вона може з'явитися вище від рамки чи бути вбудованою в неї. Якщо мітки немає чи є вона пробілом (« «) або порожньою («»), зображується тільки рамка.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – « «). Атрибут *label* виводиться як поле у верхньому лівому куті стовпця, розміщеного в рамці.

Приклад:

```
:boxed_column{
  …
}
```

ROW

Ряд – Row: Аналогічний стовпцеві, але його поля розташовані горизонтально в порядку розташування в *DCL*-файлі.

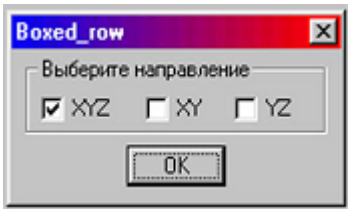
Атрибути

Ряд без рамки не має додаткових атрибутів, крім стандартних атрибутів компонування.

Приклад:

```
:row{
  …
}
```

BOXED_ROW



Ряд у рамці – Boxed_row: Ряд, що має намальовану навколо нього рамку. Якщо в рядах рамки є мітка, вона може з'явитися вище від рамки чи бути вбудованою в неї. Якщо мітки немає або вона є пробілом (« «) чи порожнью («»), зображується тільки рамка.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – « «). Атрибут *label* виводиться як поле у верхньому лівому куті ряду, розміщеного у рамці.

Приклад:

```
:boxed_row{
  …
}
```

RADIO_COLUMN

Стовпець вибору – Radio_column: Стовпець, що містить поля кнопок вибору. За один раз можна вибрати тільки одну з кнопок. Це фіксований набір взаємовиключних альтернатив. На відміну від звичайних стовпців, стовпці вибору можуть виконувати дії.

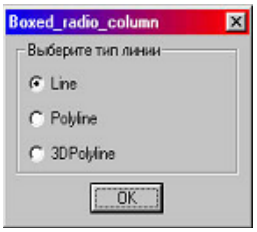
Атрибути

Value: Значення – рядок, узятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:radio_column {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

BOXED_RADIO_COLUMN



Стовпець вибору в рамці – Boxed_radio_column: Стовпець вибору, що має намальовану навколо нього рамку. Мітка трактується аналогічно мітці стовпця в рамці.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – « «). Атрибут *label* виводиться як поле у верхньому лівому куті стовпця, розміщеного в рамці.

Value: Значення – рядок, узятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:boxed_radio_column {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

RADIO_ROW

Ряд вибору – Radio_row: Аналогічний ряду вибору і містить поля кнопок вибору, але за один раз можна вибрати тільки одну кнопку. Ряд вибору може виконувати дії.

Атрибути

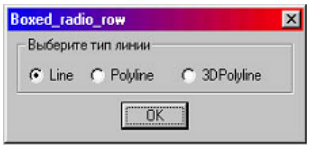
Value: Значення – рядок, узятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:radio_row {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
```

```
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

BOXED_RADIO_ROW



Ряд вибору в рамці – Boxed_radio_row: Ряд вибору, розміщений у рамці. Мітка трактується так само, як і мітка стовпця в рамці.

Атрибути

Label: Значення – рядок, узятий у лапки (за замовчуванням – « «). Атрибут *label* виводиться як поле у верхньому лівому куті ряду, розміщеного в рамці.

Value: Значення – рядок, взятий у лапки, що містить значення ключа (*key*) обраної поточної кнопки вибору.

Приклад:

```
:boxed_radio_row {
    :radio_button{ key="Lin";label = "Line";}
    :radio_button{ key="Plin";label = "Polyline";}
    :radio_button{ key="3DPlin";label = "3DPolyline";}
}
```

До активних полів також відносяться функції що визначаються кнопками виконання.

Ok_only. Аналогічна кнопці *OK* у вікні попереджень. Ключ *key* кнопки *OK* має значення «accept».

Ok_cancel. Команда, яка викликає *OK* і *Cancel*, що є стандартною комбінацією для діалогових вікон. Ключ *key* кнопки скасування має значення «cancel».

Ok_cancel_help. Групу *ok_cancel* скомбіновано зі стандартною кнопкою *Help*. Ключ *key* кнопки допомоги має значення «help».

5.1.4 Декоративні й інформаційні поля

IMAGE

Зображення – Image: Прямокутник, усередині якого відображається векторне зображення.

Атрибути

Color: Колір фону малюнка, визначений як номер кольору AutoCAD чи одне із символьних імен (за замовчуванням – 7):

dialog_line – поточний колір діалогового вікна;

dialog_forground – поточний колір символів діалогового вікна (для тексту);

dialog_background – поточний колір фону діалогового вікна;

graphic_background – поточний колір фону графічного екрана AutoCAD (як правило, аналог 0);

black – колір AutoCAD (чорний) на чорному фоні зображується як білий.

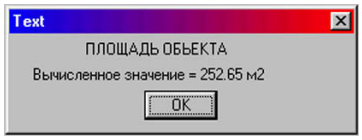
Aspect_ratio: Відношення ширини зображення до його висоти (ширина, ділена на висоту). Якщо значення дорівнює 0 (0.0), то поле зміщується під розміри зображення. Можливе значення з плаваючою точкою (значення за замовчуванням немає).

Полю зображення слід присвоїти точні значення атрибутів *width* і *height* чи один із цих атрибутів і значення *aspect_ratio*.

Приклад:

:image { key = "comp1"; color = 7; width = 30; aspect_ratio = 0.7;}

TEXT



Напис – Text: Текстовий рядок, який використовують для виведення заголовка поля та діалогового вікна чи з інформаційною метою.

Атрибути

Label: Відображуваний на екрані текст. Значення – рядок, узятий у лапки, (значення за замовчуванням немає). Під час компонування поля напису його ширина більшою від значення атрибута *width*, визначеного в *DCL*-файлі, чи ширини, обумовленої атрибутом *label*, якщо його задано. Якщо жодного з них не задано, видається повідомлення про помилку.

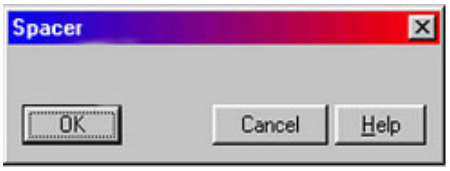
Value: Аналогічно *label* визначає рядок, виведений у поле напису, але не впливає на компонування полів. Якщо повідомлення є незмінним, варто визначити атрибут *label* і не визначати *width* і *value*. Інакше визначається атрибут *value* й атрибуту *width* присвоюється досить велике значення. Після виведення діалогового вікна на екран не можна змінити розмір його полів; якщо під час виведення функції *set_tile* напису присвоєно більш довге значення ніж ширина вікна напис буде скорочено.

Is_bold: Можливі значення: *true* чи *false* (за замовчуванням – *false*). Якщо *true*, напис виводиться напівжирним шрифтом.

Приклад:

:text {label=" New/Work";} :text {label="" ;} :text {label="Old";}

SPACER



Розділювач – Spacer: Порожнє поле. Використовують тільки з метою компонування, він впливає на розмір і розташування суміжних полів.

Атрибути

Немає.

Приклад:

spacer;

або **:spacer{width=20;};**

5.2 Функції AutoLISP для керування діалоговим вікном

Опис діалогового вікна мовою DCL є статичною картиною форми вікна. Усі дії під час діалогу та після закриття вікна визначає керуюча програма виклику діалогового вікна й обробки результатів діалогу. У посібнику розглянуто керування діалоговими вікнами тільки через AutoLISP. Спочатку описано застосовані для цього функції AutoLISP, потім розглянуто особливості їх використання в керуючих програмах. Такі керуючі програми можуть містити не тільки рядки керування діалогом, але й виконувати визначені прикладні задачі користувача, однак більш доцільно функції керування діалоговими вікнами оформляти у вигляді окремих підпрограм, викликаних із прикладної задачі. У цьому разі полегшується режим налагодження діалогу та з'являється можливість використання функцій в інших прикладних програмах.

5.2.1. Відкриття й закриття *DCL*-файлів

(load-dialog імені-файлу)

Завантажує зазначений *DCL*-файл. Повертає ціле значення (*dcl_id*), яке використовується в наступних викликах функцій *new_dialog* і *unload_dialog*. В ім'я файлу не слід установлювати розширення.

(unload-dialog dcl_id)

Вивантажує зазначений *DCL*-файл. Завжди повертає *nil*.

5.2.2. Відкриття й закриття діалогових вікон

(new_dialog ім'я-вікна dcl_id [[дія] точка])

Починає керування діалоговим вікном, виводить його на екран і може визначити дію за замовчуванням. Аргумент імені-вікна є рядком і визначає діалогове вікно, а аргумент *dcl_id* визначає *DCL*-файл (це значення визначається викликом функції *load_dialog*). Додаток має викликати функцію *new_dialog* перед викликом *start_dialog*. Усі установки, такі, як присвоєння полям значень, створення зображень чи списків для полів списку та зв'язування дій з полями (через виклики функції *action_tile*), мають відбуватися після виклику функції *new_dialog* і перед викликом *start_dialog*. Дія за замовчуванням виконується після указання користувачем активного поля, що не має присвоєного функцією *action_tile* чи визначеного в *DCL*-файлі, функції виклику з поверненням. У разі успішного завершення функції (*new_dialog*) повертає *t*, у противному разі – *nil*.

Аргумент дії, який має бути заданий, якщо визначено аргумент «точка», є рядком, що містить вираз AutoLISP для використання як дії за замовчуванням. Якщо аргумент дії визначати не потрібно, варто присвоїти йому порожній рядок (««). Якщо наявний аргумент «точка», то він є списком (*X Y*) точки розташування діалогового вікна на екрані. Точка зазвичай визначає положення діалогового вікна при його першій ініціалізації. Координати положення вікна слід задавати після виклику функції *done_dialog* для повторного відкриття вікна. Це дозволяє відкривати вікна заново, в тому місці, де їх було закрито.

Якщо точка визначена як (-1-1), діалогове вікно відкриється в центрі графічного екрана AutoCAD.

Завжди потрібно перевіряти повернуте функцією *new_dialog* значення. Виклик функції *start_dialog*, якщо функція *new_dialog* повернула значення помилки, може призвести до непередбачених наслідків.

(start_dialog)

Починає діалог у діалоговому вікні. Діалогове вікно має бути спершу ініціалізоване попереднім викликом функції *new_dialog*. Воно залишається активним, поки вираз чи діюча функція виклику з поверненням не викликають функцію *done_dialog*; зазвичай виклик *done_dialog* пов'язаний з полем, що має ключ «accept» (кнопка *OK* чи).

Функція *start_dialog* не має аргументів. Вона повертає аргумент «стан», переданий у функцію *done_dialog*. Значенням за замовчуванням є «1», якщо користувач указав кнопку ; «0» – якщо користувач указав кнопку *Cancel*; «-1», якщо вікна були закриті викликом функції *term_dialog*. Але якщо функція *done_dialog* установила стан у значення більше 1, функція *start_dialog* поверне значення, зміст якого визначається додатком.

(done_dialog [стан])

Завершує діалог у діалоговому вікні та забирає його з екрана. Функція має викликатися з виразу чи дії, функції виклику з поверненням. Ця функція також повертає поточне положення (*X Y*) діалогового вікна. Якщо функція з поверненням визначена для клавіш із значенням ключа «accept» чи «done», функція виклику з поверненням обов'язково має викликати функцію *done_dialog*. Якщо цього не зробити, користувач може не вийти з діалогового вікна. Якщо з цими клавішами не позв'язана функція виклику з поверненням і використовуються стандартні клавіші виходу, AutoCAD обробляє їх автоматично і набуває значення 1 (або визначеного додатком значення), інакше функція *start_dialog* поверне значення 0, що аналогічно скасуванню вікна.

Аргумент «стану» не є обов'язковим. Якщо він визначений, має бути позитивним цілим, яке функція *start_dialog* повертає як 1 для *OK* і як 0 – для *Cancel*. Функція *done_dialog* повертає список координат точки (*X Y*), яка визначає положення діалогового вікна, що залишив користувач. Цей список можна використовувати в наступному виклику функції *new_dialog* для повторного відкриття діалогового вікна в обраному користувачем місці.

(term_dialog)

Завершує діалог у всіх активних у цей момент діалогових вікнах, як у випадку переривання користувачем. Якщо при відкритих *DCL*-файлах виконання додатка переривається, AutoCAD автоматично викликає функцію *term_dialog*. Цю функцію використовують переважно для закриття вкладених діалогових вікон. Функція *term_dialog* завжди повертає *nil*.

5.2.3. Ініціалізація вираження дії та функцій виклику з поверненням

(action_tile ключ вираження_дії)

Присвоює дію, що буде виконуватися після вибору користувачем визначеного поля. Аргумент «ключ» є ім'ям поля, яке буде викликати дію. Дія, присвоєна функцією *action_tile*, заміщує дію за замовчуванням діалогового вікна чи атрибут поля *action*, якщо вони визначені в *DCL*-файлі.

Аргументи «ключ» і «вираження_дії» є рядками. Аргумент «вираження_дії» обчислюється після вибору поля. «Вираження_дії» може звертатися до поточного значення поля (атрибут поля *value*), визначеного як *$value*; до його імені, визначеного як *$key*; до даних, установлених функцією *client_data_tile*, визначених як *$data*; до коду причини виклику, визначеного як *$x* і *$y*.

5.2.4. Обробка – полів і атрибутів

(mode_tile ключ режим)

Установлює режим для даного поля. Аргумент «ключ» визначає поле. Аргумент «режим» є цілим числом, його значення показано в табл. 3.

Таблиця 3

Значення	Результат аргументу
0	Поле ввімкнено (доступне)
1	Поле вимкнено (недоступне)
2	Вибрати поле
3	Підсвітити вміст текстового поля
4	Увімкнути/Вимкнути підсвічування зображення

Для аргументу «режим» можна визначити тільки одне ціле значення. Аргумент «ключ» є рядком.

(get_attr ключ атрибут)

Запитує значення *DCL* зазначеного атрибута. Аргумент «ключ» визначає поле, аргумент «атрибут» – ім'я атрибута в *DCL*-описі поля. Значення повертається як установлене в описі поля; воно не відображає зміни стану поля, що можуть відбутися після введення користувачем чи виклику функції *set_tile*.

Функція *get_attr* повертає значення атрибута як рядок. Аргументи «ключ» і «атрибут» є рядками.

(get_tile ключ)

Запитує значення зазначеного поля. Аргумент «ключ» визначає поле. Функція *get_tile* повертає значення поля як рядок. Аргумент «ключ» є рядком.

(set_tile ключ значення)

Установлює «значення» для зазначеного поля. Аргумент «ключ» визначає поле, аргумент «значення» – присвоює значення. Усі аргументи є рядками.

5.2.5. Задання полів списків і списків, що розкриваються

(start_list ключ [операція] індекс])

Запускає обробку зазначеного поля чи списку, що розкривається, визначеним аргументом «ключ». Аргумент операції є цілим числом із такими можливими значеннями (табл. 4).

Таблиця 4

Операція	Результат
1	Змінити обраний вміст списку
2	Додати новий пункт у список
3	Замінити список на новий (за замовчуванням)

Аргумент «індекс» ігнорується, якщо функція *start_dialog* не викликається з кодом зміни 1, тоді індекс визначає пункт списку для зміни наступним викликом функції *add_list*. Значення «індекс» починається з нуля. Аргументи «операція» й «індекс» не обов'язкові. Якщо не визначений аргумент «операція», він за замовчуванням встановлюється в 3. Якщо не визначені «операція» та «індекс», «індекс» за замовчуванням встановлюється в 0. Аргумент «ключ» є рядком.

(add_list елемент)

Додає заданий рядок у поточний список чи замінює пункт списку на елемент. Аргумент «елемент» є рядком. Він завершує обробку поточного списку.

(end_list)

Вихід з команд задання списків.

5.2.6. Створення зображень

(dimx_tile ключ)

Функції повертають розміри поля в одиницях діалогового вікна, які використовують функції *vector_image*, *fill_image* і *slide_image*, котрі вимагають задання абсолютних координат. Аргумент «ключ» визначає поле. Координати повертаються як максимально дозволені в поле. Оскільки координати відраховуються від нуля, функції повертають значення на 1 менше, ніж розміри за *X*- чи *Y*-напрямком. Функція *dimx_tile* повертає ширину поля, функція *dimy_tile* – його висоту. Для обох функцій аргумент «ключ» є рядком.

(start_image ключ)

Запускає процес створення зазначеного зображення в поле, визначене аргументом «ключ», який є рядком.

(vector_image x1 y1 x2 y2 колір)

Креслить вектор на поточному активному зображенні (відкритому функцією *start_image*) із точки (*x1 y1*) у точку (*x2 y2*). Параметр «колір» визначає номер кольору AutoCAD чи один із логічних кольорів (табл. 5).

Таблиця 5

Номер кольору	Мнемоніка ADI	Значення
-2	BGLCOLOR	Колір фону графічного екрана AutoCAD
-15	DBGCOLOR	Колір фону діалогового вікна
-16	DFGCOLOR	Колір діалогового вікна (для тексту)
-18	LINECOLOR	Колір лінії діалогового вікна

Початок (0 0) знаходиться в лівому верхньому куті зображення. Можна одержати координати правого нижнього кута зображення, викликавши функцію розмірів *dimx_tile*.

(fill_image x1 y1 x2 y2 колір)

Малює зафарбований прямокутник на активному зображенні.

Аргументи аналогічні функції *vector_image*.

(slide_image x1 y1 x2 y2 ім'я_слайда)

Відображає слайд AutoCAD на поточному активному зображенні. Слайд може бути не лише окремим, а й бібліотечними (*.slb). У разі виклику з бібліотеки спочатку вказується ім'я бібліотеки, потім у круглих дужках – ім'я слайда. Перший кут слайда, його точка вставки, визначається як (*x1 y1*), другий кут – (*x2 y2*). Початок (0 0) знаходиться в лівому верхньому куті зображення. Завершує створення поточного зображення.

(end_image)

Вихід з команд створення зображень.

5.2.7. Дані, пов'язані з програмним додатком

(client_data_tile ключ клієнти-дані)

Пов'язує керовані додатком дані з полем, визначеним аргументом «ключ». Аргумент «ключ» є рядком. Визначені додатком дані задає аргумент, «клієнти-дані» також є рядком. Вираження дії може посилатися на рядок, визначений як *$data*.

5.3 Схеми викликів функцій керування

5.3.1. Виклик з файлу *exsample.dcl*

Приклад виклику та керування простим діалоговим вікном:

```
exsample: dialog {label = «Приклад діалогового вікна»;
: text {label = «Діалогове вікно» ; }
ok_only;
 }
```

Указаний текст знаходиться у файлі *exsample.dcl*. Функція AutoLISP, яка керує появою цього вікна, може мати такий вигляд:

```
(defun showalert ( / dcl_id)
  (setq dcl_id (load_dialog «exsample.dcl»))      ;Завантажити DCL-файл
  (if (not (new_dialog «exsample» dcl_id)) ;Ініціалізувати діалог
  (exit))                  ;(вийти, якщо не працює)
  (action_tile)            ;Пов'язати вираження дії
  «accept»                 ;з ключем кнопки OK
  «(done_dialog)»)              ;Закінчити діалог, якщо натиснуто кнопку OK
  (start_dialog)»)              ;Вивести діалогове вікно
  (unload_dialog dcl_id)        ;Вивантажити DCL-файл
```

)

Після виклику функції *start_dialog* діалогове вікно стає активним, поки користувач не підсвітить поле (зазвичай, кнопку), пов’язане з викликом вираження *done_dialog*. Виклик функції *action_tile* установлює зв'язок між полем (у цьому прикладі з кнопкою *OK*, ключ якої має значення «accept») і вираженням дії. Саме через цей виклик функція *done_dialog* з’являється всередині виклику функції *action_tile* і перед викликом функції *start_dialog*. По суті, всі дії до функції *start_dialog* є коригувальними. Більш складні вікна потребують більшої кількості додаткових операторів між функціями *start_dialog* і *unload_dialog*, але послідовність викликів буде незмінною.

Наведений приклад демонструє послідовність викликів функцій:

1. Завантаження *DCL*-файлу функцією *load_dialog*.
2. Виклик функції *new_dialog* для виведення окремого діалогового вікна на графічний екран AutoCAD. Важливо перевіряти повернене функцією *new_dialog* значення. Виклик функції *start_dialog*, якщо функція *new_dialog* повернула помилку, може призвести до непередбачених результатів.
3. Ініціалізація діалогового вікна та встановлення значень полів, списків і зображень, якщо необхідно. На цьому етапі викликаються функції *set_tile* і *mode_tile* для встановлення стану полів та їхніх значень; *start_list*, *add_list*, *end_list* – для полів списків; *start_image*, *vector_image*, *fill_image*, *slide_image*, *end_image* – для зображень. Аналогічно можна використовувати функцію *action_tile* для зв'язку з діалоговим вікном і його компонентами.
4. Виклик функції *start_dialog* передає керування діалоговому вікну, в яке користувач може вводити дані.
5. Процес уведення даних користувачем (виклики з поверненням). На цьому етапі використовують функції *get_tile*, *get_attr*, *set_tile*, *mode_tile*.
6. Указання користувачем клавіші виходу викликає функцію *done_dialog*, яка, у свою чергу, повертає функцію *start_dialog*. Далі відбувається вивантаження *DCL*-файлу, забезпечуване функцією *unload_dialog.*

Якщо діалогове вікно (функцію *start_dialog* викликано) є активним, то не можна викликати деякі функції AutoLISP, що змінюють екран, який не має змінюватися, поки на ньому зображується діалогове вікно.

Якщо необхідно ввести дані в режимі графічного екрана (наприклад, вибрати точку чи примітив), слід тимчасово закрити діалогове вікно, викликавши функцію *done_dialog*, – графічний екран стане доступним. Після виконання вибору виконується повторний виклик вікна.

Перелік функцій AutoCAD, заборонених до виклику при відкритому діалоговому вікні:

1. Запити і команди AutoCAD: *command*, *osnap*.
2. Функції введення користувачем: *getint, getreal, getstring, getpoint, getcorner, getdist, getangle, getorient, getword*.
3. Функції керування екраном: *prompt, menucmd, redraw, graphscr, textscr, textpage*.

4. Графічні функції низького рівня: *grclear, grdraw, grread, grtext, grvacs*.
5. Функція набору вибору *ssget*.
6. Функції керування примітивами: *entmod, entmake, entdel, entsel, nentsel, entpd*.

5.3.2. Вирази дії

Для визначення наслідків вибору визначеного поля діалогового вікна варто зв'язати вираження AutoLISP із цим полем за допомогою виклику функції *action_tile*. Усередині дії часто буває необхідно одержати доступ до атрибутів *DCL*-файлу. Це забезпечують функції *get_attr* і *get_tile*: функція *get_attr* видає значення, збережені в *DCL*-файлі, а *get_tile* – поточні значення. Значення, пов'язані з обраним полем, виходять автоматично.

Вираз дії AutoLISP одержує доступ до змінних, котрі подано в табл. 6, й описують обране поле та його стан під час дії. Імена змінних зарезервовані, можна вживати тільки їх значення (вони мають сенс тільки тоді, коли доступні вираженню дії).

Таблиця 6

Змінна	Призначення	Примітка
$key	Атрибут *key* обраного поля	Застосовують до всіх дій
$value	Рядок поточного значення поля, схожий на рядок вікна редагування, «0» чи «1» для перемикача	Застосовують до всіх дій. Якщо поле є полем списку (чи списку, що розкривається) і не містить жодного пункту, змінна *$value* буде мати значення *nil*
$data	Дані, встановлені відразу після виклику *new_dialog* через виклик *client_data_tile* і керовані додатком	Застосовують до всіх дій. Не має значення доти, поки додаток не установить його викликом функції *client_data_tile*
$reason	Код причини, що повідомляє, який вибір користувача викликав дію. Використовується в полях *edit_box, list_box, image_button* і *slider*	Визначає причину дії. Її значення визначене для будь-якого виду дії, але його варто перевіряти, якщо дія пов'язана з полями *edit_box, list_box; image_button* і *slider*
$x	Координата *X* вибору *image_button*	Представляє координати *X* і *Y* точки, в якій користувач вибрав поле *image_button*
$y	Координата *Y* вибору *image_button*	Для інших цілей змінна не має значення. Координата *X* знаходиться в інтервалі, який повертає функція *dimx_tile*, координата *Y* – в інтервалі, котрий повертає функціея *dimy_tile*

Змінна *new_tile* встановлюється в значення атрибута *key* обраного поля, тобто в «*edit_*». Змінну *$key* часто використовують усередині функцій для виконання дій, присвоєних декількома різними полями.

5.3.3. Установлення полів списків і списків, що розкриваються

При встановленні значень для поля чи списку, що розкривається, необхідно виконати таку послідовність операцій:

(start_list), (add_list) і (end_list)

Список можна змінити після його створення. Функцію *mapcar* слід використовувати для переведення списку з AutoLISP у відображуване поле списку:

```
(start_list «selections» операція)
(mapcar ‘add_list newnames)
(end_list)
```

5.3.4.Обробка значень списків

Під час оброблення списків значення поля *list_box* може містити початкові пробіли, особливо якщо обрано кілька пунктів. Спочатку варто перетворити значення до цілого, використовуючи функцію *atoi* чи *read*.

Приклад: Передбачається, що список «*justone*» дозволяє вибрати тільки один пункт списку. Оскільки функція *atoi* повертає 0, якщо рядок порожній, чи коли в ньому міститься «0», то спочатку варто перевірити, чи є рядок порожнім.

```
(setq index (get_tile «justone»))
(cond
((/=index ««)
(= 2 (atoi index)) ; це третій пункт
...
            )
        )
```

Списки, що розкриваються, не дозволяють виконувати множинний вибір. Якщо в поле списку можна вибрати одночасно декілька пунктів, програма має не тільки виконати перетворення, але і перебрати безліч значень у рядку.

Приклад: Функція *mk_list* повертає список, що містить тільки обрані користувачем поля з вихідного списку *displist*. Змінна *displist* визначена як глобальна. Передбачається, що функція *mk_list* викликається з поточним значенням *$value* поля списку.

```
(defun mk_list (readlist / count item retlist)
(setq count 1)
(while (setq item (read readlist))
(setq retlist (cons (nth item displist) retlist))
(while (and (/= « «(substr readlist count 1))
(+ « «(substr readlist count 1)))
(setq count (1+ count))
)
(setq readlist (substr readlist count)))
(reverse retlist))
```

5.3.5.Обробка зображень

Послідовність викликів для створення поля зображення та клавіші зображення аналогічні послідовності оброблення списку. Функція *start_image* починає створення зображення, функція *end_image* закінчує його. Однак визначення результату малювання здійснюється використанням декількох різних функцій, а не заданням різних аргументів:

– *vector_image* – креслить вектор (одна пряма лінія) на поточному зображенні;

– *fill_image* – малює зафарбований прямокутник;

– *slide_image* – відображає слайд AutoCAD.

Ці функції вимагають визначення абсолютних координат. Щоб задати їх коректно, варто знати розміри поля чи зображення клавіші зображення. Оскільки розміри полів визначаються під час компонування і забезпечують одержання ширини та висоти поля: *dimx_tile* і *dimy_tile*. Початок поля (0 0) завжди знаходиться у верхньому куті.

Приклад: Потрібно зафарбувати червоним кольором поле зображення «cur_color»:

```
(setq width (dimx_tile «cur_color»)
    height (dimy_tile «cur_color»))
(start_image «cur_color»)
(fill_image «cur_color»)
(fill_image 0 0 width height 1) ; 1 – червоний колір AutoCAD
(end_image)
```

Далі намалюємо навколо поля межу червоного кольору, а не зафарбований прямокутник (вектори креслять проти годинникової стрілки):

```
(setq width (dimx_tile «border»)
    height (dimy_tile «border»))
(start_image «border»)
(vector_image 0 0 0 height 1)
(vector_image 0 height width height 1)
(vector_image width height width 1)
(vector_image width 0 0 0 1)
(end_image)
```

Зобразимо зафарбоване зображення і накреслимо на ньому вертикальну лінію:

```
(setq width (dimx_tile «striple»)
height (dimy_tile «striple»))
(start_image «striple»)
(fill_image 0 0 width height 3); 3 – зелений колір AutoCAD
(setq x(/width 2.0)
```

(vector_image x 0 height 4); 4 – фіолетовий колір AutoCAD
(end_image)

Відображені за допомогою функції *slide_image* слайди можуть бути окремими слайдами-файлами (*.sld*). Розширення вказувати не потрібно.

Припустимо, що необхідно відобразити слайд *tppview.sld*:

```
(setq x (dimx_tile «view»)
    y (dimy_tile «view»))
(start_image «view»)
(slide_image 0 x y «topview»)
(end_image)
```

Вектори слайда часто малюють білим кольором, що є кольором фону зображення за замовчуванням. Якщо слайд після виведення не видно, треба змінити атрибут *color* на значення *graphic_background*.

5.3.6. Уведення кнопки зображення

Кнопку зображення можна обробляти як звичайну кнопку, тобто при її указанні викликається відповідна дія. Однак є можливість визначення кнопку так, що дія буде залежати від місця вказання в зоні кнопки. Для цього поле функції кнопки має одержати координати вказівки. Координати задаються в діапазоні розміру зображення і повертаються функціями визначення розмірів.

Приклад: Припустимо, що кнопка зображення має дві прямокутні області різного кольору. Потрібно визначити, яку з областей указав користувач. Якщо кнопку поділено навпіл горизонтальною лінією, варто перевіряти тільки один розмір:

```
(action_tile «image_set» «(pick_shade» $key $value $y)»)
...
(defun pick_shade (key val y)
(setq threshold (/dimy_tile key) 2))
(if (yy threshold)
(setq result «Світлий»)
(setq result «Темний»))
)
```

5.3.7. Обробка ковзних шкал

Дію ковзної шкали перевіряє код причини. Як приклад покажемо базову схему функції керування ковзною шкалою. Вона викликається з пов’язаного з нею вираження дії. Поле *slider_info* ця функція використовує для виведення в десятковій формі поточного значення стану шкали і є текстовим полем, що дозволляє користувачу керувати станом ковзної шкали, безпосередньо вводячи значення. Якщо користувач увів значення стану ковзної шкали, то:

```
(action_tile «my_slider» «(slider_action $value $reason)»)
(action_tile «my_info» «(ebox_action $value $reason)»)
```

```
...
(defun slider_action (val why)
     (if (or (=why 2) (=why 1))(set_tile «slider_info» val) )
)
(defun ebox_action (val why)
     (if (or (= why 2) (= why 1)) (set_tile «myslider» val) )
)
```

5.4 Приклади створення діалогових вікон

1. Діалогове вікно з використанням **Edit_box**.

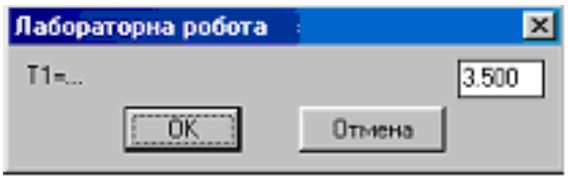


Рисунок 8 – Просте діалогове вікно

Файл dial.dcl

```
dialog_lr: dialog { label = "Лабораторна робота" ;
  :edit_box{ label="T1=..."; edit_width=5; edit_limit=5; key="tt1";}
    ok_cancel ;
}
```

Файл dial.lsp

```
(defun c:dialog ()
     (setq rfile (load_dialog "dial.dcl"))
     (new_dialog "dialog_lr" rfile)
     (setq t1 3.5)
     (set_tile "tt1" (rtos t1 2 3))
     (action_tile "tt1" "(progn (setq t1 (atof $value)))")
     (action_tile "accept" "(setq pozdlg (done_dialog 1))")
     (action_tile "cancel" "(done_dialog 0)")
     (setq rslt (start_dialog))
     (print t1)(princ)
)
```

2. Діалогове вікно з визначенням активних полів в рядку

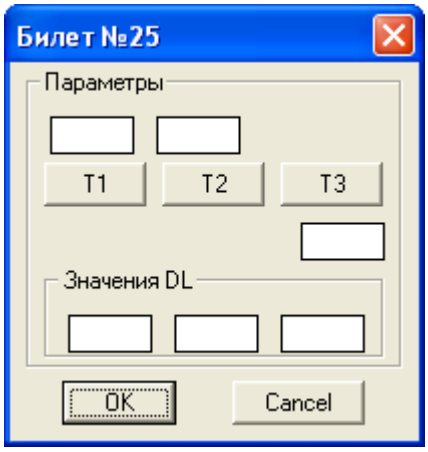


Рисунок 9 – Групуванням активних полів в рядок

Файл DCL

```
dial_1:dialog{label="Билет №25";
 :boxed_column{label="Параметры";
  :row{
    :edit_box{edit_width=5;edit_limit=5;key="E1";}
    :edit_box{edit_width=5;edit_limit=5;key="E2";}
    :spacer{width=10;}
   }
   :row{
      :button{label="T1";key="T1";}
      :button{label="T2";key="T2";}
      :button{label="T3";key="T3";}
  }
  :row{
    :spacer{width=20;}
    :edit_box{edit_width=5;edit_limit=5;key="E3";}
  }
  :boxed_row{label="Значения DL";
    :edit_box{edit_width=5;edit_limit=5;key="E4";}
    :edit_box{edit_width=5;edit_limit=5;key="E5";}
    :edit_box{edit_width=5;edit_limit=5;key="E6";}
  }
 } // boxed_column
ok_cancel;
}
```

Файл AutoLISP

```
(defun c:LS_25 ()
(setq rfile (load_dialog "H:\\Bl_25.dcl"))
(new_dialog "dial" rfile)
(setq E1 1.0)(set_tile "E1" (rtos e1))
(setq E2 2.0)(set_tile "E2" (rtos e2))
(setq E3 3.0)(set_tile "E3" (rtos e3))
(setq E4 4.0)(set_tile "E4" (rtos e4))
(setq E5 5.0)(set_tile "E5" (rtos e5))
(setq E6 6.0)(set_tile "E6" (rtos e6))

(action_tile "E1" "(progn (setq E1 (atof $value)))")
(action_tile "E2" "(progn (setq E2 (atof $value)))")
(action_tile "E3" "(progn (setq E3 (atof $value)))")
(action_tile "E4" "(progn (setq E4 (atof $value)))")
(action_tile "E5" "(progn (setq E5 (atof $value)))")
(action_tile "E6" "(progn (setq E6 (atof $value)))")

(action_tile "T1" "(T1)")
(action_tile "T2" "(T2)")
(action_tile "T3" "(T3)")

(action_tile "accept" "(setq pozdlg (done_dialog
1))")
(action_tile "cancel" "(done_dialog 0)")
(setq rslt (start_dialog))
(princ)
)
;---------------------------------------
(defun t1()  (alert "Кнопка T1"))
(defun t2()  (alert "Кнопка T2"))
(defun t3()  (alert "Кнопка T3"))
```

3. Діалогове вікно з визначенням активних полів в колонці

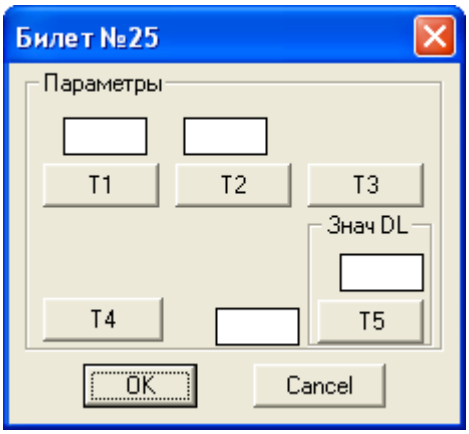


Рисунок 10 – Групуванням активних полів в колонку

Файл DCL

```
dial_2:dialog{label="Билет №25";
 :boxed_column{label="Параметры";
  :row{
    :edit_box{edit_width=5;edit_limit=5;key="E1";}
    :edit_box{edit_width=5;edit_limit=5;key="E2";}
    :spacer{width=10;}
  }
  :row{
    :button{label="T1";key="T1";}
    :button{label="T2";key="T2";}
    :button{label="T3";key="T3";}
  }
  :row{
    :column{
       :spacer{width=10;}
       :button{label="T4";key="T4";}
      }
      :column{
       :spacer{width=10;}
       :edit_box{edit_width=5;edit_limit=5;key="E4";}
      }
     :column{label="Знач DL";
     :edit_box{edit_width=5;edit_limit=5;key="E5";}
     :button{label="T5";key="T5";}
    }
   } // row
 } //  boxed_column
ok_cancel;
}
```

Файл AutoLISP

```
 (defun c:LS_25 ()
(setq rfile (load_dialog "H:\\Bl_25.dcl"))
(new_dialog "dial" rfile)
(setq E1 1.0)(set_tile "E1" (rtos e1))
(setq E2 2.0)(set_tile "E2" (rtos e2))
(setq E3 3.0)(set_tile "E3" (rtos e3))
(setq E4 4.0)(set_tile "E4" (rtos e4))

(action_tile "E1" "(progn (setq E1 (atof $value)))")
(action_tile "E2" "(progn (setq E2 (atof $value)))")
(action_tile "E3" "(progn (setq E3 (atof $value)))")
(action_tile "E4" "(progn (setq E4 (atof $value)))")

(action_tile "T1" "(T1)")
(action_tile "T2" "(T2)")
(action_tile "T3" "(T3)")
(action_tile "T4" "(T4)")
(action_tile "T5" "(T5)")

(action_tile "accept" "(setq pozdlg (done_dialog
1))")
(action_tile "cancel" "(done_dialog 0)")
(setq rslt (start_dialog))
(princ)
)
;---------------------------------------
(defun t1()  (alert "Кнопка T1"))
(defun t2()  (alert "Кнопка T2"))
(defun t3()  (alert "Кнопка T3"))
(defun t4()  (alert "Кнопка T4"))
(defun t5()  (alert "Кнопка T5"))
```

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. *Руководство* пользователя системы AutoCAD.
2. Автоматизація графічно-конструкторських робіт у процесі проектування хімічного устаткування в системі AutoCAD: Навч.посіб. / В.Ю. Щербина, О.С. Сахаров, О.В. Гондлях, В.І. Сівецький. – К.: ІВЦ „Видавництво „Політехніка», 2003. – 152с.: іл
3. Автоматизоване проектування черв'ячного устаткування: Навч. посіб. / В.І. Сівецький, В.Ю. Щербина – К.: ІВЦ «Видавництво «Політехніка»«, 2005. – 184с.: іл
4. САПР. Програмний комплекс АПРОКС в розрахунках машин та апаратів хімічних виробництв: Навч. посіб. / О.В. Гондлях, О.С. Сахаров, В.І. Сівецький, В.Ю. Щербина, Р.М. Пашинський, А.О. Чемерис. – К.: ТОВ
5. САПР. Інтегрована система моделювання технологічних процесів і розрахунку обладнання хімічної промисловості: Навч. посіб. / О.С.Сахаров, В.Ю.Щербина, О.В. Гондлях, В.І. Сівецький. – К.: ТОВ «Поліграф Консалтинг», 2006. – 156с.: іл
6. САПР. Чисельне моделювання гідромеханічних процесів та НДС суцільних середовищ при термосилових навантаженнях: Навч. посіб. / О.С. Сахаров, В.Ю. Щербина, В.І. Сівецький, О.В. Гондлях – К.: ТОВ «Поліграф Консалтинг», 2007. – 180с.: іл
7. *Кречко Ю. А*. AutoCAD: программирование и адаптация. – М.: ДИАЛОГ-МИФИ, 1995. – 240 с.
8. Сайт „Компьютерное конструирование» http://tomskcad.city.tomsk.net/CAD/CAD.htm.
9. Список функций Visual Lisp http://www.cad.dp.ua/stats/vlisp.php
10. Кудрявцев Е.М. AutoLISP. Программирование в AutoCAD 14 / Кудрявцев Е.М. -М.: «ДМК», 1999 - 368 с, ил.
11. Энкарначчо Ж., Шлехтендаль Э. Автоматизированное проектирование: Основные понятия и архитектура систем: Пер. с англ. – М.: Радио и связь, 1986. – 288 с.
12. 12.Бугрименко Г. А., Лямке В. Н., Шейбонене Э.-К. С. Автоматизация конструирования на ПЭВМ с использованием системы AutoCAD. – М: Машиностроение, 1993. – 336 с.
13. Хювенен Э., Сеппянен Й. Мир Лиспа: В 2 т. – М.: Мир, 1990.
14. *Гладков С. А*. Програмирование на языке Автолисп в системе САПР AutoCAD. – М.: ДИАЛОГ-МИФИ, 1991. – 96 с.
15. *Бугрименко А. С.* Автоматизация конструирования на ЭВМ с использованием системы AutoCAD. – М.: Машиностроение, 1993. – 336 с.
16. *Батбаро В. А., Заблоцкий Д. В., Кмеллер М. И.* AutoCAD. Полезные рецепты – М.: Радио и связь, 1994. – 208 с.
17. *Крюков А. П., Родионов А. Я.* Программирование на языке R-Lisp –М.: *Радио* и связь, 1991. – 192 с.
18. *Paulson J. Layer* upon layer of goodies with A'LISP // CAD user. – 1989. – N6. – P. 80.

Електронне мережне навчальне видання

Щербина Валерій Юрійович

КОНСТРУКТОРСЬКЕ ПРОЕКТУВАННЯ ОБЛАДНАННЯ КОНСПЕКТ ЛЕКЦІЙ

для студентів,
які навчаються за спеціальністю 133 «Галузеве машинобудування»,
спеціалізації «Інжинирінг, комп'ютерне моделювання та проектування обладнання виробництв полімерних і будівельних матеріалів та виробів»

Комп'ютерна правка та верстка – *авторські*